

**Masterarbeit im Fach Informatik**

# **Two-Dimensional Warping for Image Recognition**

Der Fakultät für  
Mathematik, Informatik und Naturwissenschaften der  
RHEINISCH-WESTFÄLISCHEN TECHNISCHEN HOCHSCHULE AACHEN

Lehrstuhl für Informatik 6  
Prof. Dr.-Ing. H. Ney

vorgelegt von:  
Harald Hanselmann  
Matrikelnummer 252400

Gutachter:  
Prof. Dr.-Ing. H. Ney  
Prof. Dr. B. Leibe

Betreuer:  
Dipl.-Inform. Philippe Dreuw

April 2011



# Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Textauszüge und Grafiken, die sinngemäß oder wörtlich aus veröffentlichten Schriften entnommen wurden, sind durch Referenzen gekennzeichnet.

Aachen, im April 2011

Harald Hanselmann



# Abstract

The task of image recognition is very challenging due to many intra-class variations. Especially the example of face recognition offers many challenges such as illumination, facial expressions, occlusions or different poses. Many different approaches have been proposed such as analyzing the sub-space spanned by face images, feature matching or three-dimensional models. However the approach of two-dimensional warping is particularly suited for this task. It defines a similarity measure between images that is very tolerant to local deformations and can be used for nearest-neighbor classification, which allows mug-shot recognition.

In this thesis novel two-dimensional warping algorithms for image recognition within a nearest-neighbor classification framework are proposed. The new algorithms maintain full dependencies in both dimensions defined by a two-dimensional grid. This approach allows the enforcement of geometric constraints such as Sakoe constraints. This is implemented without sacrificing efficiency by relaxing the general two-dimensional warping criterion. During the alignment of columns a lookahead inspired by Tree-Serial Dynamic Programming (TSDP) is used.

The developed algorithms are evaluated and compared to state-of-the-art warping methods on the task of face recognition. For this purpose the AR-Face and the CMU-PIE database are considered. To raise the level of difficulty, the former is altered by introducing artificial rotations. Experiments on both databases show the competitiveness of the novel algorithms and the effectiveness of the lookahead.



# Danksagung

Zunächst möchte ich mich bei Herrn Prof. Dr.-Ing. Hermann Ney für die Möglichkeit bedanken, meine Masterarbeit zu so einem interessanten Thema zu verfassen. Weiterhin möchte ich mich bei Herrn Prof. Dr. Bastian Leibe für das Zweitgutachten dieser Arbeit bedanken.

Darüber hinaus danke ich Philippe Dreuw für die hervorragende Betreuung. Auch bei dem Rest der Image-Gruppe des Lehrstuhls für Informatik 6 der RWTH Aachen möchte ich mich für jegliche Unterstützung bedanken, insbesondere bei Jens Forster, Tobias Gass und Leonid Pishchulin.

Ein ganz besonderer Dank geht an meine Eltern, die mich während des gesamten Studiums immer unterstützt haben.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Classification Framework</b>	<b>5</b>
2.1	Nearest-Neighbor Classifier . . . . .	5
2.2	Features . . . . .	6
<b>3</b>	<b>Warping Algorithms</b>	<b>7</b>
3.1	Warping in General . . . . .	7
3.2	Sakoe constraints . . . . .	9
3.3	Zero-Order Warping . . . . .	12
3.4	Pseudo Two-Dimensional Warping . . . . .	13
3.5	Tree-Serial Dynamic Programming . . . . .	15
3.6	Sequential Tree-Reweighted Message Passing . . . . .	17
<b>4</b>	<b>Two-Dimensional Warping Algorithms</b>	<b>19</b>
4.1	Constrained TSDP . . . . .	19
4.2	Two-Level Dynamic Programming . . . . .	21
4.2.1	Dynamic Programming . . . . .	22
4.2.2	Initialization . . . . .	24
4.2.3	Column-level . . . . .	25
4.2.4	Complexity . . . . .	25
4.3	Restriction to $u$ -component . . . . .	26
4.4	Local-Best . . . . .	27
4.5	Strip-Restriction . . . . .	28
4.6	Lookahead . . . . .	31
4.7	Aligned Lookahead . . . . .	33
4.8	Comparison of the Warping Algorithms . . . . .	34
4.8.1	Search Criterion . . . . .	35
4.8.2	Complexity . . . . .	36
<b>5</b>	<b>Practical Aspects</b>	<b>39</b>
5.1	Distance Caching . . . . .	39
5.2	Distance Thresholding . . . . .	39
5.3	Nearest Neighbor Pruning . . . . .	40

<b>6 Databases</b>	<b>41</b>
6.1 Rotated-45 AR-Face . . . . .	41
6.2 CMU-PIE . . . . .	41
<b>7 Results</b>	<b>43</b>
7.1 Zero-Order Warping . . . . .	44
7.2 Pseudo Two-Dimensional Warping . . . . .	45
7.3 Tree-Serial Dynamic Programming . . . . .	48
7.4 Sequential Tree-Reweighted Message Passing . . . . .	48
7.5 Constrained Tree-Serial Dynamic Programming . . . . .	50
7.6 Two-Level Dynamic Programming . . . . .	54
7.6.1 Lookahead . . . . .	55
7.6.2 Restriction to $u$ -component . . . . .	61
7.6.3 Local-best selection strategy . . . . .	62
7.6.4 Strip-Restriction . . . . .	62
7.6.5 Lookahead weight . . . . .	64
7.6.6 Representative . . . . .	64
7.7 Nearest Neighbor Pruning . . . . .	65
7.8 Summary of the Results . . . . .	66
7.8.1 Rotated-45 AR-Face . . . . .	66
7.8.2 CMU-PIE . . . . .	66
<b>8 Conclusion</b>	<b>69</b>
<b>A Appendix</b>	<b>71</b>
A.1 Software . . . . .	71
<b>List of Figures</b>	<b>73</b>
<b>List of Tables</b>	<b>75</b>
<b>Bibliography</b>	<b>77</b>

# Chapter 1

## Introduction

The term image recognition as a sub-field of pattern recognition describes the task of automatically identifying an object or multiple objects within a given image. The image containing the object to identify is usually called test image, the images used for training are called reference or training images. Identifying in this context means that given a set of possible classes, the correct one that the object belongs to has to be determined. In order to do this, the between-class variations that characterize the differences between the classes are utilized. This however is problematic, since also within the objects of the same class many variations can occur. For instance color values of an object can vary significantly by applying different illuminations, especially under natural conditions. Also rotations, translations or scalings can lead to different appearances of the same object. These and other within-class variations are what makes the topic of image recognition challenging. On the other hand considering the wide area of applications it is worth the effort. Image recognition is used for optical character recognition, texture error detection or driver assistance systems in the automobile industry. The example used in this thesis is the task of face recognition, due to its many intra-class variations. There is a wide variety of facial expressions, from anger to happiness, from smiling to shouting. Also the effects of aging, different hair-styles and even plastic surgery can be an issue. Therefore, face recognition poses an especially challenging task, leading to a very active area in the image recognition research community [Gross & Shi<sup>+</sup> 01, Zhao & Chellappa<sup>+</sup> 03, Tan & Chen<sup>+</sup> 06, Pinto & DiCarlo<sup>+</sup> 09].

Consequently over the past several years, many different algorithms have been proposed to solve the problem of face recognition. First of all, there are subspace-projection methods, such as Linear Discriminant Analysis (LDA) [Belhumeur & Hespanha<sup>+</sup> 97, Martinez & Kak 01], the Independent Component Analysis (ICA) [Shakhnarovich & Moghaddam 04] or the Principal Component Analysis (PCA) [Turk & Pentland 91]. These methods create a subspace from a set of training images into which a test image is projected for classification. Since they are created from face images, these subspaces are often referred to as the face space. The approach using the PCA is thereby from special relevance. It is one of the best known and most used methods for face recognition. Since for this method the face space is spanned by the eigenvectors of the covariance matrix of the training data the approach is also known as the eigenface approach [Turk & Pentland 91]. There are also several extensions to this such as applying the PCA on a sub-pattern level [Tan & Chen 05]. Similar to the eigenfaces for PCA,

the basis vectors of the subspace found by LDA are called fisherfaces [Martinez & Kak 01].

Other approaches focus on the used feature space. There are methods based on Local Binary Patterns (LBP) [Ahonen & Hadid<sup>+</sup> 04], a local feature that is computed by accumulating a histogram about the color value differences in a neighborhood. Other approaches use features computed by Discrete Cosine Transformation (DCT) [Ekenel & Stiefelhagen 06], or SIFT/SURF feature descriptors [Bicego & Lagorio<sup>+</sup> 06, Dreuw & Steingrube<sup>+</sup> 09] that are based on histograms of gradients. The latter will be explained in more detail in Section 2.2. Elastic Bunch Graph Matching (EBGM) [Wiskott & Fellous<sup>+</sup> 97] uses Gabor filters as features but also allows for small local deformations.

Also more and more research is done on 3D face recognition [Scheenstra & Ruifrok<sup>+</sup> 05]. For once there is a number of methods where the 2D recognition is supported by creating a 3D model from a training set of face images and synthesizing new reference images in different poses or illuminations. In [Banz & Vetter 03] a morphable 3D model is trained using an optical flow algorithm, in [Zhang & Samaras 06] a model is constructed using spherical harmonics from just one training image and in [Zhang & Gao<sup>+</sup> 08] two training images, one frontal face and one profile are used by minimizing the surface roughness while forcing equality on some predefined constraining points in both training images. Other 3D recognition algorithms try to use the 3D information more directly. In [Xu & Li<sup>+</sup> 09], features are extracted on both, the image containing the intensity values and the depth map. The used features are Gabor filters that are reduced in dimensionality using LDA. The features from both inputs are then fused into a single classifier using AdaBoost. In [Berretti & Del Bimbo<sup>+</sup> 10] iso-geodesic stripes are computed on the face using the depth information. A similarity measure between two images for classification is then obtained by comparing the mutual displacement of each pair of stripes within one image.

Another approach is image warping. The idea is to define a similarity measure between two images by how well one image can be transformed to the other. Several methods using warping have already been proposed. Some are based on Markov Random Fields. In [Arashloo & Kittler 09] an energy function is defined using the distance between two pixels as data term and a relative constraint on neighboring pixels as pair-wise term. For efficiency reasons this function is first used block-wise, the results are then used to narrow down the search in the exact image. A similar approach with improved structural constraints and occlusion handling was proposed in [Gass & Dreuw<sup>+</sup> 10]. Other approaches compute direct solutions to a given warping model. The work in [Hua & Akbarzadeh 09] searches pixel-wise for the lowest matching costs in an absolute neighborhood. In [Keysers & Deselaers<sup>+</sup> 07] a model was proposed where each pixel is allowed to be warped by certain absolute distance. In [Gass & Pishchulin<sup>+</sup> 11] more sophisticated models are presented that rely on relative constraints and local dependencies between pixels.

It has been shown that using geometric constraints on warpings lead to an improved recognition performance [Gass & Pishchulin<sup>+</sup> 11]. However, two-dimensional constraints can only be applied when two-dimensional dependencies are maintained during the computation. Most warping algorithms maintain the dependencies only in one dimension, since two-dimensional

warping in general is NP-complete [Keysers & Unger 03]. In this thesis, two new warping models are introduced with the goal to incorporate dependencies between pixels in a two-dimensional neighborhood. The latter allows to fully apply geometric constraints on the warpings, such as the Sakoe constraints [Uchida & Sakoe 98]. The first model is a modification of Tree-Serial Dynamic Programming (TSDP) [Pishchulin 10] that dissolves the independencies of the column alignments that are present in TSDP. The second model divides the optimization into a global and column level and will be extended by a lookahead inspired also by TSDP. The new models are evaluated on two databases with an emphasis on a special subset of the AR-Face database [Martinez & Benavente 98] and compared to state-of-the-art warping methods. It will be shown that the new algorithms are competitive regarding error rates, warping scores and complexity.

The thesis is structured as follows. In Chapter 2 the nearest neighbor based classification framework is explained, followed by a theoretical background regarding two-dimensional warping and a review of the warping methods already in use in Chapter 3. In Chapter 4 the new algorithms are introduced. Chapter 5 contains some notions on practical issues. Chapter 6 introduces the database on which in Chapter 7 the evaluation of the warping methods is performed. Finally, a conclusion will be given in Chapter 8.



# Chapter 2

## Classification Framework

This chapter describes the general framework in which the warping algorithms are used. It is based on a nearest-neighbor classifier.

### 2.1 Nearest-Neighbor Classifier

The framework uses a simple 1-nearest neighbor classifier for the recognition. A given test object that needs to be classified is compared to all objects in the reference database. Each reference object is assigned a score that measures the distance between the test and the reference object according to some distance function. The result of the classification is then determined by selecting the class of the reference object that is assigned the lowest score.

In our case, we have given a test image  $X = \{x_{ij}\}$  with the width  $I$  and the height  $J$  and are searching for the reference image  $R = \{r_{uv}\}$  with the width  $U$  and the height  $V$  that has the lowest distance to  $X$ . This decision rule can be defined as follows.

$$\hat{c} = \arg \min_c \left\{ \min_n d(X, R_{cn}) \right\} \quad (2.1)$$

The optimal class  $\hat{c}$  is found by minimizing over the distances  $d(., .)$  between all images  $R_{c1}, \dots, R_{cN}$  of a class  $c$  and the test image  $X$ . For the distance function there are several options. In this thesis the  $l_1$  norm, i.e. the absolute distance is used. It is defined in Equation 2.2, where  $D$  is the dimension of the feature vector describing the pixels. For simple gray-values, the dimension is equal to one but higher dimensional feature vectors will be used for classification (cf. Section 2.2).

$$d(x_{ij}, r_{uv}) = \sum_{d=1}^D |x_{ij}^d - r_{uv}^d| \quad (2.2)$$

For some tasks it is helpful to include the local context when the distance is computed. The distance between pixel  $(i, j)$  and  $(u, v)$  then describes the distance of two  $n \times n$  patches of which  $(i, j)$  and  $(u, v)$  are the corresponding centers.

However, using the distance function directly is not very flexible. Translations, rotations and other deformations that can occur within a class are not considered. To compensate this drawback, two-dimensional warping is used to model these within class deformations. Instead of applying the distance function directly to the input images, a matching between the two images is computed along the way according to a specific warping model. The warping models used in this thesis will be explained in Chapter 3 and Chapter 4.

## 2.2 Features

Instead of using the raw image data as features, feature descriptors are extracted from the data in a preprocessing step. There are many possible descriptors available, such as Sobel-[Jähne 02], DCT-[Ekenel & Stiefelhagen 06] or SIFT-Features [Lowe 99]. For this thesis, the latter feature descriptors have been chosen, since they have the important property of being rotation invariant and lead to good error rates [Dreuw & Steingrube<sup>+</sup> 09].

The SIFT-Feature descriptor analyzes the gradients around the pixel to be described. The different gradients of  $4 \times 4$ -areas are accumulated to histograms of the eight main orientations. This is done for  $4 \times 4$  such areas. The result is a 128-dimensional feature vector. The invariance regarding rotation is achieved by aligning the gradients according to the main orientation of the image. The descriptors also offer invariance to illumination by normalizing the vector to unit length. This operation compensates for different contrasts caused by different illumination.

In [Dreuw & Steingrube<sup>+</sup> 09] also an up-right version of the SIFT-descriptor (U-SIFT) was tested on image recognition tasks, which performed slightly better than the normal SIFT-features. For this descriptor the alignment of the gradients according to the main orientation is omitted. However, this sacrifices the rotation invariance of the descriptor.

When an image is processed, the SIFT-Feature descriptors are extracted at each position of the 2D-Grid. This results in a high dimensional feature vector for each pixel. To reduce the computational overhead these vectors are reduced to 30-dimensional vectors using Principal Component Analysis [Ke & Sukthankar 04].

# Chapter 3

## Warping Algorithms

In this chapter, after introducing Two-Dimensional Warping in general, the already existing warping algorithms are reviewed.

### 3.1 Warping in General

Given the test image  $X = \{x_{ij}\}$  and the reference image  $R = \{r_{uv}\}$ , the warping problem defines the search for a mapping from each pixel in the test image to a pixel in the reference image according to some specific warping model. This warping mapping is defined in Equation 3.1.

$$(i, j) \rightarrow w_{ij} = (u_{ij}, v_{ij}) \quad (3.1)$$

Each pixel  $(i, j)$  in the test image is assigned a pixel  $(u_{ij}, v_{ij})$  in the reference image. The latter is abbreviated with  $w_{ij}$ . Combining all such  $w_{ij}$  for each test pixel results in a set  $\{w_{ij}\}$  that captures the complete warping. In order to be able to comprise the structures in the test image and avoid arbitrary mappings, dependencies between the warping of each pixel and the warping of the neighboring pixels are defined, i.e. the warping of pixel  $(i, j)$  depends on the warpings of its direct neighbors. Since the image is a two-dimensional grid, this leads to two-dimensional dependencies (cf. Figure 3.1). To capture all dependencies, for each pixel two predecessors are considered, one in the horizontal and one in the vertical direction. These dependencies are called first-order dependencies, since in both dimensions one predecessor is considered.

$$\min_{\{w_{ij}\}} \left\{ \sum_{i,j} [d_{ij}(w_{ij}) + T(w_{i-1,j}, w_{i,j-1}, w_{ij})] \right\} \quad (3.2)$$

To find the best warping mapping with two-dimensional dependencies, the general warping criterion is defined in Equation 3.2. It is composed of the sum over the distances between the pixel  $(i, j)$  and its mapping  $w_{ij}$  plus the two-dimensional warping penalty  $T(w_{i-1,j}, w_{i,j-1}, w_{ij})$ .

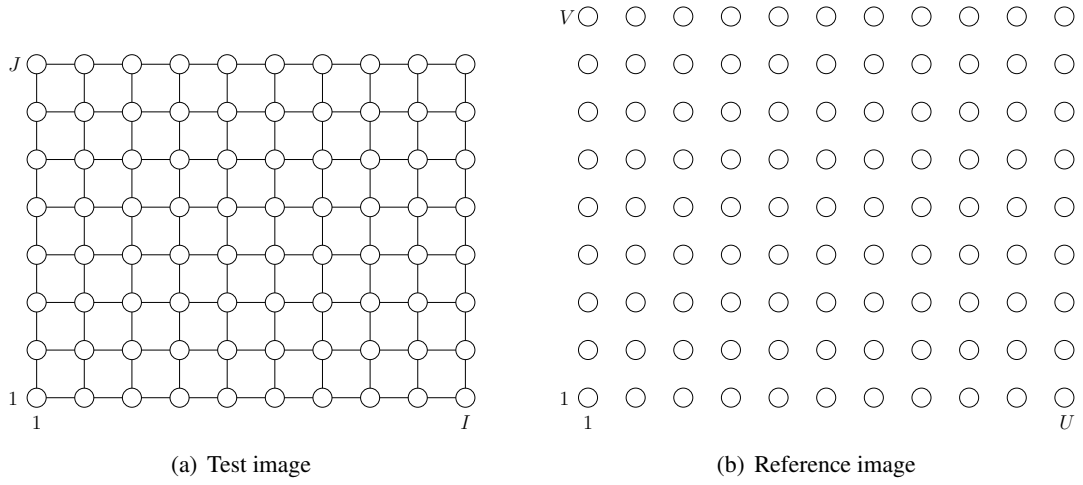


Figure 3.1: The test image is a two-dimensional grid. For each pixel a warping mapping in the reference image is searched, depending on the warping mappings of the direct neighbors. [Ney & Dreuw<sup>+</sup> 10]

The latter is a function that penalizes large deviations in the mappings of  $(i, j)$  and its predecessors in both dimensions. It can be decomposed into a vertical and horizontal component [Ney & Dreuw<sup>+</sup> 10].

$$T(w_{i-1,j}, w_{i,j-1}, w_{ij}) = T_H(w_{i-1,j}, w_{ij}) + T_V(w_{i,j-1}, w_{ij}) \quad (3.3)$$

In this thesis, the Euclidean distance is used as penalty function (cf. Equation 3.4 and Equation 3.5). By adding the vector  $(1, 0)$  and  $(0, 1)$  respectively the warping is penalty free, when the the warpings  $w_{ij}$ ,  $w_{i-1,j}$  and  $w_{i,j-1}$  have the same relative positions in the reference image, as  $(i, j)$ ,  $(i-1, j)$  and  $(i, j-1)$  have in the test image. Otherwise the penalty increases, the further the mappings are apart. The contribution of the warping penalty is regulated by the weight factor  $\alpha$ .

$$T_H(w_{i-1,j}, w_{ij}) = \alpha \cdot \|w_{i-1,j} + (1, 0) - w_{ij}\|_2 \quad (3.4)$$

$$= \alpha \cdot \sqrt{(u_{i-1,j} + 1 - u_{ij})^2 + (v_{i-1,j} + 0 - v_{ij})^2}$$

$$T_V(w_{i,j-1}, w_{ij}) = \alpha \cdot \|w_{i,j-1} + (0, 1) - w_{ij}\|_2 \quad (3.5)$$

$$= \alpha \cdot \sqrt{(u_{i,j-1} + 0 - u_{ij})^2 + (v_{i,j-1} + 1 - v_{ij})^2}$$

Unfortunately, it has been proven that solving the minimization problem in Equation 3.2 is NP-complete [Keysers & Unger 03]. This leaves two options to tackle the warping problem

and find efficient solutions. The first option is to approximate a solution to Equation 3.2. The second option is to relax the general warping criterion and solve the relaxation optimally. In Chapter 3 and Chapter 4 we will see examples for both approaches.

## 3.2 Sakoe constraints

As we have seen in the previous section, the pixels in the 2D grid have vertical and horizontal dependencies. That can be exploited to enforce relative geometric constraints on the warping mapping. If two pixels  $p = (i, j)$  and  $q = (i', j')$  are neighbors in the test image, it is likely that they belong to the same geometric structure. To maintain these structures, also their mappings  $w_p$  and  $w_q$  should be close together.

In this thesis, the Sakoe constraints [Uchida & Sakoe 98] are used for this purpose. They are composed of two parts, the monotonicity and the continuity, which then again is divided into a horizontal and a vertical part.

$$0 \leq u_{ij} - u_{i-1,j} \leq 2 \quad (3.6)$$

$$-1 \leq v_{ij} - v_{i-1,j} \leq +1 \quad (3.7)$$

$$0 \leq v_{ij} - v_{i,j-1} \leq 2 \quad (3.8)$$

$$-1 \leq u_{ij} - u_{i,j-1} \leq +1 \quad (3.9)$$

The monotonicity constraints, horizontal (cf. Equation 3.6) and vertical (cf. Equation 3.8), enforce that the position of the mapping for pixel  $(i, j)$  is always equal or greater than the position of the mapping for pixels  $(i - 1, j)$  and  $(i, j - 1)$ . The continuity constraints (cf. Equation 3.7 and Equation 3.9) allow a deviation of at most one pixel. This is illustrated in Figure 3.2 for the horizontal constraints and Figure 3.3 for the vertical constraints. In both cases, the black pixel in the test image is mapped onto the black pixel in the reference image allowing nine possible mappings for the predecessor of the black pixel.

It is easy to see that the Sakoe constraints force pixels that are close in the test image to be also close in the reference image. However, there are some restrictions. The monotonicity does not allow mirroring of the objects. Additionally, only rotations of less or equal to 90 degrees, clockwise or counter-clockwise, can be modeled (cf. Figure 3.4 and Figure 3.5). The horizontal constraints tolerate the predecessor of a pixel  $(i, j)$  to be mapped at most to the position under (or over respectively) of the position  $(i, j)$  itself is mapped to. This corresponds to a rotation of exactly 90 degrees. Due to the limited number of pixel skips allowed by the Sakoe constraints, also scalings are limited.

A positive side effect of applying the Sakoe constraints is a reduction of runtime for algorithms optimizing over possible predecessors of a pixel mapping. This is due to the reduced number of predecessor candidates. Given a reference image of the size  $64 \times 64$ , without any constraints, there are 4096 such candidates. However, when using the Sakoe constraints this number is reduced to nine.

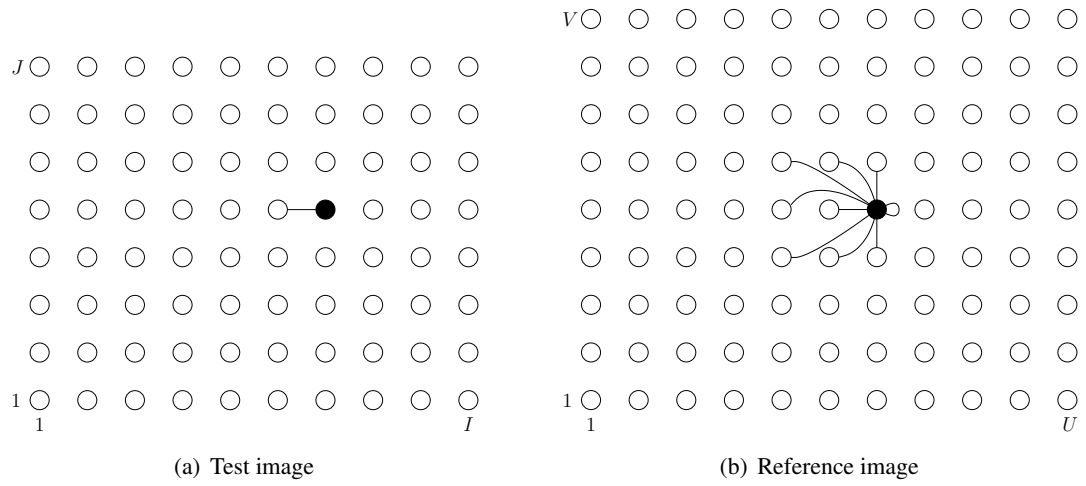


Figure 3.2: Horizontal Sakoe constraints and the nine possible mappings for the predecessor of the black pixel. [Ney & Dreuw<sup>+</sup> 10]

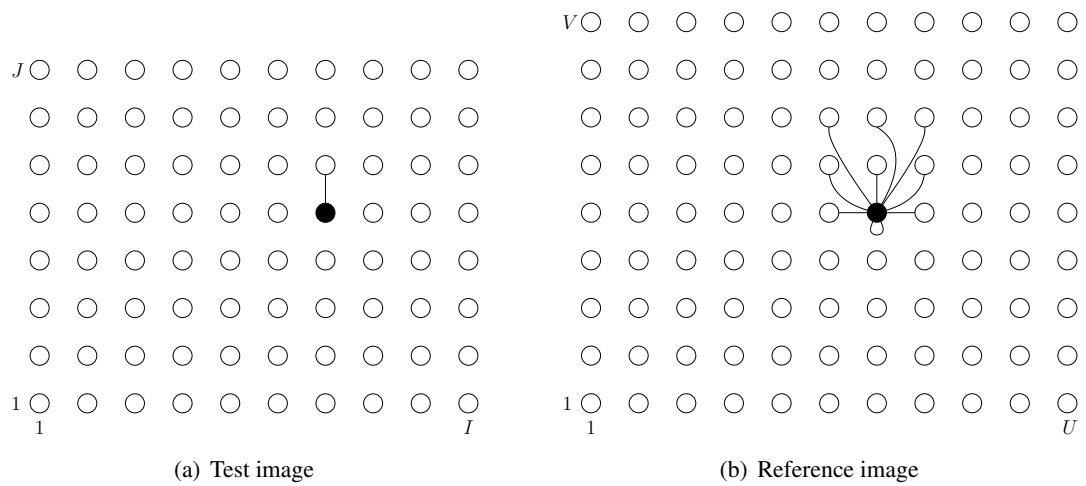


Figure 3.3: Vertical Sakoe constraints and the nine possible mappings for the predecessor of the black pixel. [Ney & Dreuw<sup>+</sup> 10]

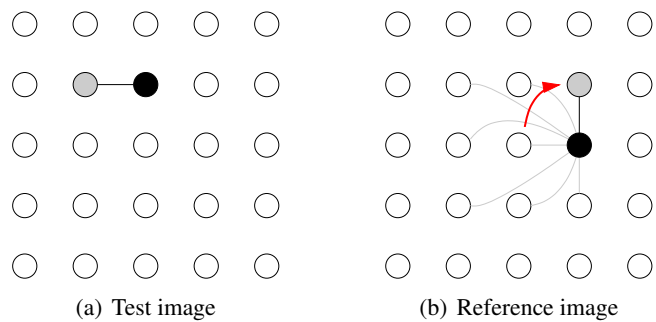


Figure 3.4: Rotation of at most 90 degrees clockwise.

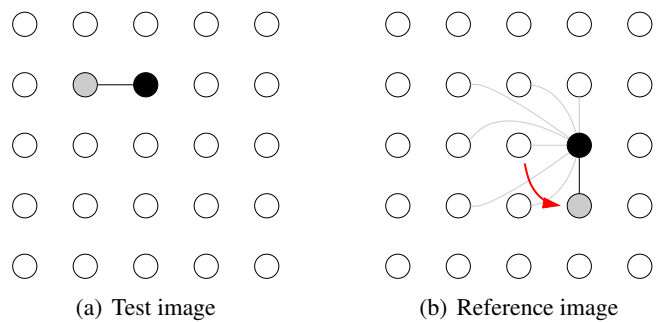


Figure 3.5: Rotation of at most 90 degrees counter-clockwise.

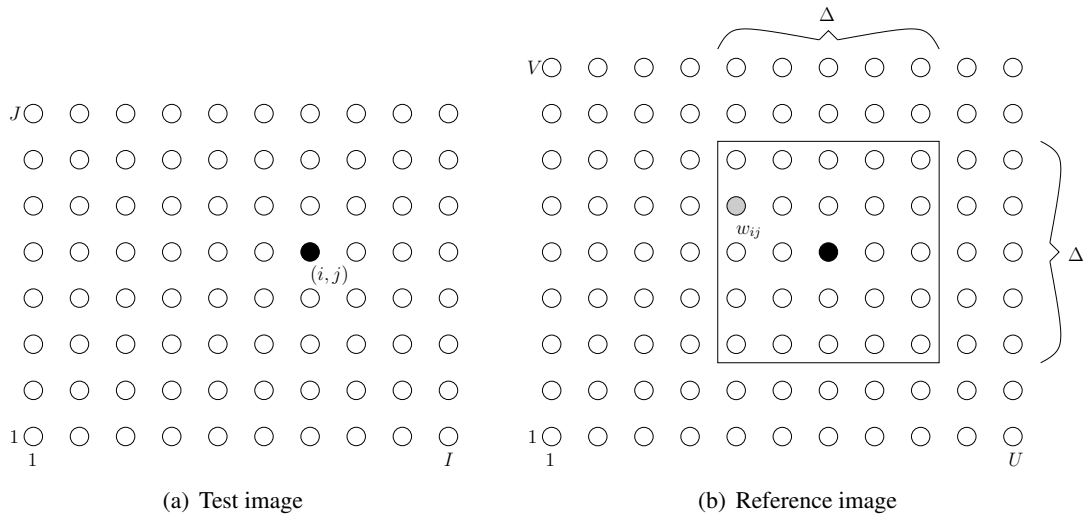


Figure 3.6: Zero-Order Warping: The mapping of pixel  $(i, j)$  is restricted only by the warprange  $\Delta$ .

### 3.3 Zero-Order Warping

The Zero-Order Warping (ZOW) [Keyzers & Deselaers<sup>+</sup> 07] is the direct implementation of the zero-order model, where no dependencies between the warpings of the pixels in the test images are considered. Since there are no first-order dependencies at all, each pixel  $(i, j)$  in the test image is mapped independently from all other pixels. For this purpose, the pixel  $(i, j)$  is compared to each pixel in the reference image and the reference pixel leading to the lowest score is selected as a fixed decision. Thereby, the range of the possible mapping candidates  $(u, v)$  is restricted by a warprange  $\Delta$ , such that each pixel can not be warped further than allowed by the warprange (cf. Equation 3.10).

$$|u - i| \leq \Delta \wedge |v - j| \leq \Delta \quad (3.10)$$

Figure 3.6 illustrates how the ZOW works. The pixel  $(i, j)$  is mapped onto the pixel  $w_{ij} = (u, v)$ , which is located within the warprange  $\Delta$  around the original location of  $(i, j)$  indicated by the black pixel.

The lack of first-order dependencies also leads to an absolute warping penalty. The penalty  $T(w_{ij})$  is determined by the absolute distance between  $(i, j)$  and  $w_{ij}$ . This already unfolds the weakness of the ZOW. Strong translations, rotations or scalings of an object can either not be compensated at all, due to a limited warprange, or will cause high penalties. The absolute constraints also make it difficult to compute good warpings for images with different dimensions. On the other hand, the ZOW is very fast. The complexity is given by  $I \cdot J \cdot \Delta^2$ , since for each pixel  $(i, j)$  an area of the size  $\Delta \times \Delta$  has to be searched for the lowest distance

of the pixel values. If the warprange is unlimited, this leads to a worst case complexity of  $I \cdot J \cdot U \cdot V$ , which is still very efficient compared to the other methods.

### 3.4 Pseudo Two-Dimensional Warping

The Pseudo Two-Dimensional Warping (P2DW) is a model that implements the first-order dependencies in a simplified form. Originally based on hidden Markov models (HMM) [Kuo & Agazzi 94, Keyser & Deselaers<sup>+</sup> 07], it combines all pixels within the same column and then maps the resulting super-pixels. This means that one column in the test image is always mapped onto another complete column in the reference image. For this mapping column-wise horizontal first-order dependencies and the monotonicity of the Sakoe constraints is enforced. Let  $u_i$  be the column that  $i$  is mapped on. The constraints on the column-wise mapping are then given by the following formula.

$$0 \leq u_i - u_{i-1} \leq 2 \quad (3.11)$$

Since the vertical dimension is reduced to one during the column mapping, the continuity constraints are not needed. On column-level, when column  $i$  is mapped onto column  $u_i$ , the best mapping of the pixels in column  $i$  onto the pixels of column  $u_i$  is computed, influenced by the vertical first-order dependencies. Here again the monotonicity of the Sakoe constraints is applied. This division of a column-wise mapping with a nested best vertical column mapping allows to divide the warping penalty function in the following way [Ney & Dreuw<sup>+</sup> 10].

$$T(w_{i-1,j}, w_{i,j-1}, w_{ij}) \quad (3.12)$$

$$= T(u_{i-1}, u_i, v_{i,j-1}, v_{ij}) \quad (3.13)$$

$$= T(u_{i-1}, u_i) + T(v_{i,j-1}, v_{ij}) \quad (3.14)$$

Thereby  $T(u_{i-1}, u_i)$  is the horizontal penalty for the transition from one column to the next, while  $T(v_{i,j-1}, v_{ij})$  is the vertical penalty within a column mapping. The division into a global and a column level is also reflected in the new optimization criterion for the P2DW [Ney & Dreuw<sup>+</sup> 10].

$$\begin{aligned} & \min_{\{w_{ij}\}} \left\{ \sum_{i,j} [d_{ij}(w_{ij}) + T(w_{i-1,j}, w_{i,j-1}, w_{ij})] \right\} \\ = & \min_{\{u_i, v_{ij}\}} \left\{ \sum_{i,j} [d_{ij}(u_i, v_{ij}) + T(u_{i-1}, u_i, v_{i,j-1}, v_{ij})] \right\} \end{aligned} \quad (3.15)$$

$$= \min_{\{u_i\}} \left\{ \sum_i \min_{\{v_{ij}\}} \left\{ \sum_j [d_{ij}(u_i, v_{ij}) + T(u_{i-1}, u_i) + T(v_{i,j-1}, v_{ij})] \right\} \right\} \quad (3.16)$$

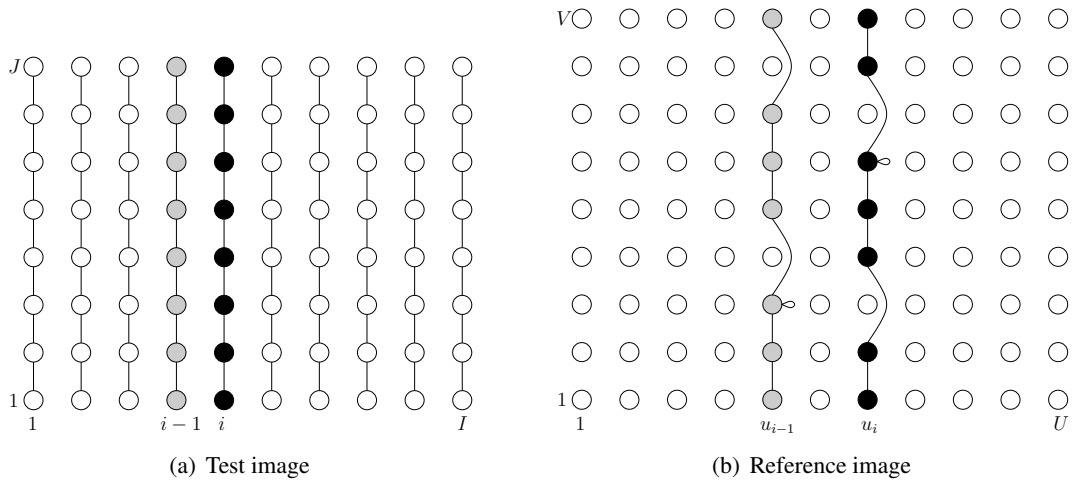


Figure 3.7: Pseudo Two-Dimensional Warping: Columns are treated as super-pixels and mapped onto other complete columns. Within the columns the vertical Sakoe constraints are enforced.

First, the inner minimization of Equation 3.16 optimizes the within column mappings, then the outer minimization optimizes over the selection of the columns. Originally, the P2DW also applied boundary constraints, such as forcing the first column of the test image to be mapped onto the first column of the reference image [Keysers & Deselaers<sup>+</sup> 07]. As shown in [Pishchulin 10], applying boundary constraints limits the warping mappings and leads to a reduced recognition performance. Therefore, no boundary constraints are used in this thesis.

Additionally, forcing columns to be mapped on complete columns significantly restricts the possible warping mappings. While translations or scalings can still be handled, rotations can not be compensated, since this would require pixels of the same column in the test image to be mapped on pixels in different columns in the reference image. For this reason, in [Pishchulin 10] a First-Order Strip Extension (FOSE) is introduced, allowing also horizontal deviations during a column mapping by a constant of  $\Delta$ .

The principle of the P2DW and the strip extension are shown in Figure 3.7 and Figure 3.8. Figure 3.7 demonstrates a possible warping of column  $i$  given the mapping of column  $i - 1$  using the P2DW. All pixels belonging to  $i$  must be mapped onto the same column in the reference image and must be mapped within the vertical monotonicity constraints. In Figure 3.8 the difference caused by the strip extension is visible. Now, also a deviation within the defined window is possible.

Since the P2DW-FOSE easily outperforms the basic P2DW [Pishchulin 10], only the P2DW-FOSE will be considered. The approach leads to a model that can be solved efficiently by dynamic programming.

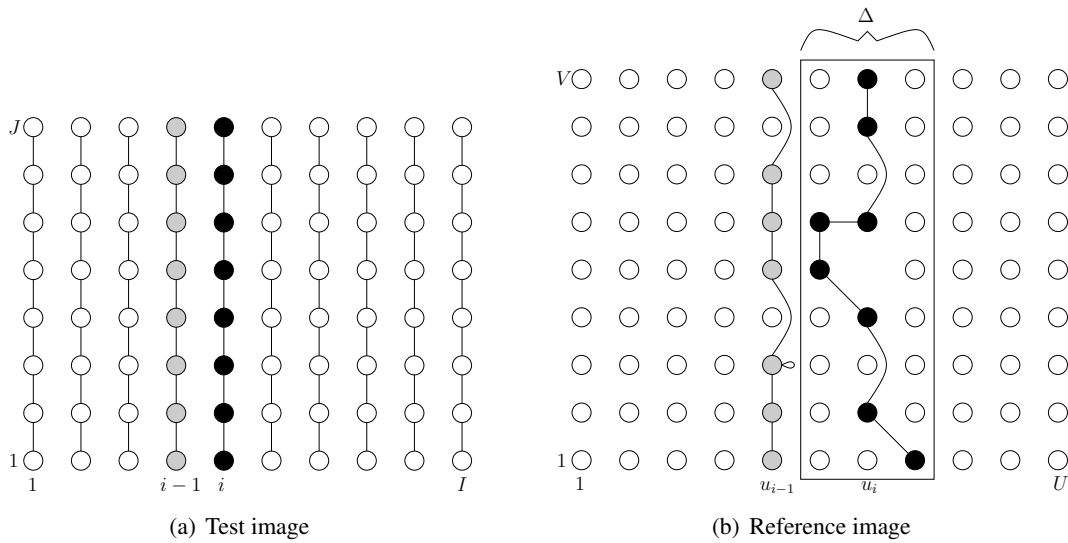


Figure 3.8: Pseudo Two-Dimensional Warping with first-order strip extension: Also deviations within a defined strip are possible.

The overall complexity is composed of the following parts.

- $3 \cdot U \cdot I$  for the global level. For each pair of test and reference column the score needs to be computed given the possible predecessors. Due to the Sakoe constraints, there are 3 predecessors possible over which the minimum has to be determined.
- $9 \cdot (2\Delta + 1) \cdot I \cdot J \cdot U \cdot V$  for the column level. To compute one column matching, a dynamic programming table of the dimension  $(2\Delta + 1) \cdot J \cdot V$  is used, where one entry holds the costs of mapping the  $j$ -th position of the test column onto the  $v$ -th position of the reference column with a given horizontal deviation. The additional factor of 9 is due to the optimization over the predecessors.

Combining complexities listed above, the total complexity of the P2DW-FOSE is given by  $3 \cdot U \cdot I + 9 \cdot (2\Delta + 1) \cdot I \cdot J \cdot U \cdot V$ .

### 3.5 Tree-Serial Dynamic Programming

The Tree-Serial Dynamic Programming (TSDP) [Mottl & Kopylov<sup>+</sup> 02] is another relaxation to the general warping criterion (cf. Equation 3.2) with the goal to maintain first-order dependencies. The idea is to decompose the 2D grid into a set of trees. The trees are constructed by discarding all vertical dependencies except for one column. This is done for each column leading to a set of  $I$  trees. Each tree is then optimized independently leading to an alignment

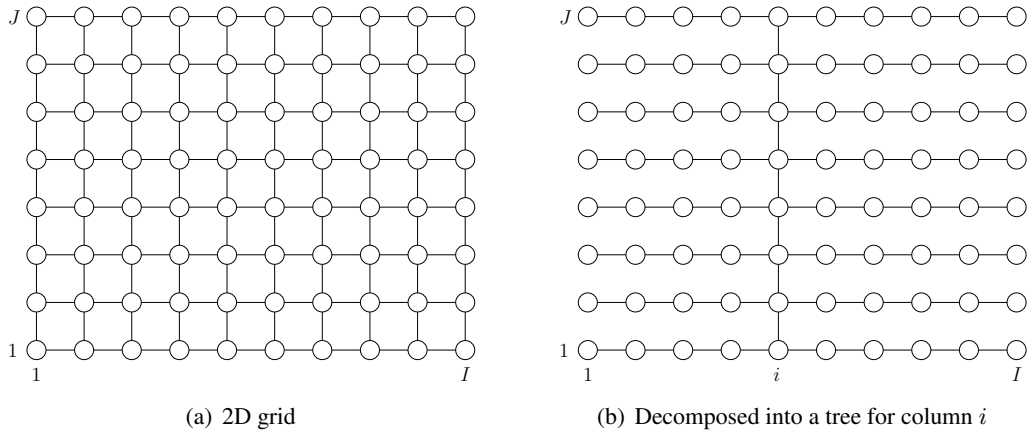


Figure 3.9: Tree-Serial Dynamic Programming: The grid is decomposed into a set of trees, which are then optimized independently. [Ney & Dreuw<sup>+</sup> 10]

for that specific column. The procedure is illustrated in Figure 3.9. When optimizing column  $i$ , the tree in Figure 3.9(b) is considered. This can be computed efficiently by dynamic programming, since all cycles are removed.

A major advantage of this approach is that the tree must not be optimized from scratch for each column. By using one forward and one backward run of dynamic programming, the branches can be optimized at once. The results are simply stored in a look-up table which is consulted when the columns are optimized. The formulas for computing these look-up tables are given in Equation 3.17 and Equation 3.18 [Pishchulin 10, Ney & Dreuw<sup>+</sup> 10].

$$H_L(i^*, j, w_{i^*j}) = \min_{w_{ij}, i < i^*} \left\{ \sum_{i < i^*} [d_{ij}(w_{ij}) + T(w_{i+1,j}, w_{ij})] \right\} \quad (3.17)$$

$$H_R(i^*, j, w_{i^*j}) = \min_{w_{ij}, i > i^*} \left\{ \sum_{i > i^*} [d_{ij}(w_{ij}) + T(w_{i-1,j}, w_{ij})] \right\} \quad (3.18)$$

While  $H_L$  is the look-up table for the branches left from the column to be optimized,  $H_R$  contains the entries for the right part. In both tables, for each column  $i^*$  and each position  $j$  in that column, there is an entry for each possible mapping  $w_{i^*j}$  that contains the minimal costs for the rest of the row. Accumulated over the complete column, these scores can be interpreted as some kind of lookahead that gives information about the approximate costs of the rest of the image. The columns are then optimized using Equation 3.19 [Pishchulin 10]. Additionally to the distance and warping penalty, now also the forward and backward scores  $H_L$  and  $H_R$  are used.

$$\min_{w_{i^*j}} \left\{ \sum_j [d_{i^*j}(w_{i^*j}) + T(w_{i^*,j-1}, w_{i^*j}) + H_L(i^*, j, w_{i^*j}) + H_R(i^*, j, w_{i^*j})] \right\} \quad (3.19)$$

In the original version by [Mottl & Kopylov<sup>+</sup> 02] the set of possible warpings  $w_{ij}$  for a pixel  $(i, j)$  was restricted by an absolute warprange similar to the warprange of ZOW. However, in [Pishchulin 10] this was changed to using relative constraints, i.e. the Sakoe constraints.

As already mentioned earlier, exploiting the fact that the tree branches can be computed by a single forward-backward run of dynamic programming leads to an efficient warping model. In more detail, the complexity is defined by the following points.

- $I \cdot J \cdot U \cdot V \cdot 9$  for the backward run in the tree branch computation.
- $I \cdot J \cdot U \cdot V \cdot 9$  for the forward run in the tree branch computation.
- $I \cdot J \cdot U \cdot V \cdot 9$  for optimizing the columns.

As result, the overall complexity is at  $I \cdot J \cdot U \cdot V \cdot 27$ .

### 3.6 Sequential Tree-Reweighted Message Passing

In contrast to the previous warping algorithms, the Constrained Sequential Tree-Reweighted Message Passing (CTRW-S) is an approximation to the general warping criterion in Equation 3.2. The criterion can be seen as energy function  $E(X, R, \{w_{ij}\})$  depending on the reference image  $R$ , the test image  $X$  and the labeling  $\{w_{ij}\}$ . To find the labeling that minimizes this function the message passing algorithm can be used. Given a graph  $G$  corresponding to the problem at hand, the message passing algorithm minimizes the energy by iteratively passing messages from node to node. In our case, the underlying graph  $G$  is defined by assigning each pixel of the test image a node that is connected to each node corresponding to a neighboring pixel. However, since this graph is cyclic, a special variant of message passing is used, the Tree-Reweighted Message Passing [M. Wainwright 02]. If  $G$  was a tree, the message passing algorithm would need two passes, one inward from the leaves to the root and one outward from the root to the leaves [Kolmogorov 06]. During one pass, messages of the following form are passed along a tree branch, where  $p$ ,  $q$  and  $r$  are nodes in the graph.

$$M_{p,q}(w_q) = \min_{w_p} \{d_p(w_p) + T_{p,q}(w_p, w_q) + M_{r,p}(w_q)\} \quad (3.20)$$

The message  $M_{p,q}(w_q)$  from  $p$  to  $q$  is sent for each possible labeling  $w_q$ , informing  $q$  about the minimal costs of the tree branch so far, in case  $q$  is mapped to  $w_q$ . This minimal cost is composed of the distance  $d_p(w_p)$  of  $p$  and  $w_p$ , the warping penalty  $T_{p,q}(w_p, w_q)$  when  $q$  is

mapped to  $w_q$  and  $p$  is mapped to  $w_p$ , and the message  $M_{r,p}(w_q)$  from  $r$ , the predecessor of  $p$ , to  $p$ , which then again holds the costs of the tree branch until  $r$ . Note that this corresponds to a forward and a backward pass of dynamic programming. The solution can then be found by minimizing over the marginals  $\Phi_p$  (cf. Equation 3.21) for each  $p$ .

$$\Phi_p(w_q) = d_p(w_p) + \sum_{q \in \mathcal{N}(p)} M_{q,p}(w_p) \quad (3.21)$$

Thereby  $\mathcal{N}(p)$  is the neighborhood of  $p$ , i.e. all adjacent nodes in the graph.

However, for two-dimensional warping  $G$  is not a tree but a cyclic graph. To apply the algorithm, the graph is decomposed into tree-like subproblems. The message passing is then applied to each sub-problem and in the next step the minimal marginals are forced to agree by node averaging [M. Wainwright 02]. By summing up the energies of the subproblems, a lower bound of the energy for the actual problem is found. In order to guarantee the property that in each iteration the lower bound does not decrease, the algorithm is altered in [Kolmogorov 06]. A total order on the nodes is introduced and the message passing and node averaging are done sequential for each node (TRW-S). For an efficient implementation, Kolmogorov also limits the set of trees to monotonic chains. In this case, the messages towards a node do not need to be updated. Therefore it is sufficient to keep only one message per edge in the memory. In [Gass & Dreuw<sup>+</sup> 10] this algorithm has been further restricted to implement the Sakoe constraints resulting in the Constrained TRW-S (CTRW-S). The complexity of the CTRW-S can be described as follows.

- $I \cdot J \cdot 9 \cdot U \cdot V$  for updating the messages for each pixel in the image.
- 2 passes per iteration, one forward and one backward.

Let  $N$  be the number of iterations, then this leads to a total complexity of  $N \cdot 2 \cdot I \cdot J \cdot 9 \cdot U \cdot V$ .

# Chapter 4

## Two-Dimensional Warping Algorithms

In this chapter, two fully two-dimensional warping algorithms will be introduced. Starting with a Sakoe constrained version of the TSDP followed by an algorithm called Two-Level Dynamic Programming and its variants and extensions.

### 4.1 Constrained TSDP

Previously we have seen the Tree-Serial Dynamic Programming (cf. Section 3.5), where for each column a tree-structure was optimized. Thereby, the horizontal and vertical dependencies were used in a sequential fashion in two steps. During the construction of the tree branches, the horizontal dependencies are used, while the vertical dependencies appeared in the alignment of the columns. However, since each column is optimized independently of the others, there are no horizontal dependencies between the columns. In the final warping, the horizontal dependencies are only considered implicitly by the use of the trees. Therefore the Sakoe constraints can only be applied partially to the TSDP, i.e. only the vertical Sakoe constraints are enforced in the final warping. The now introduced constrained TSDP (C-TSDP) is a modification to the TSDP, such that all the Sakoe constraints can be enforced.

The algorithm works in two phases. At first, the tree branches are computed with one forward and one backward run of dynamic programming. This phase is identical to the TSDP. In the second phase, the columns are aligned, i.e. for each pixel  $(i, j)$  the corresponding pixel  $w_{ij}$  in the reference image is searched. The phase begins by picking one column  $i_s$  as starting column. This first column is then aligned just as it is done in the TSDP. However, the rest of the columns is aligned differently. Since we picked one column and already fixed the alignment, the two neighboring columns  $i_s + 1$  and  $i_s - 1$  can be aligned given the alignment of column  $i_s$ . This allows to consider the horizontal dependencies and to enforce also horizontal Sakoe constraints. After the two neighbors of  $i_s$  are aligned, there is now for column  $i_s + 2$  a fixed alignment of column  $i_s + 1$  that can be used to apply the horizontal constraints. Accordingly for  $i_s - 2$  the alignment of column  $i_s - 1$  can be used. Analogously, all other columns can be processed given the alignment of their neighbors.

The method is illustrated in Figure 4.1. At first the red column  $i_s$  is aligned. Then the neighboring columns  $i_s - 1$  (green) and  $i_s + 1$  (blue) are aligned and so on.

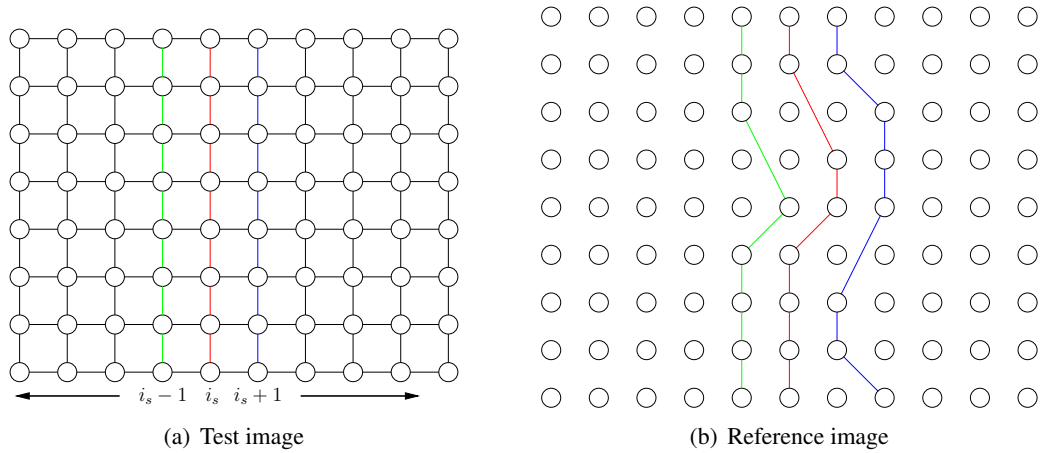


Figure 4.1: Constrained TSDP: Column  $i_s$  in the test image (a) is aligned first by finding corresponding pixels in the reference image (b). Then the neighbor columns are aligned one by one.

If we recall the formulas for the TSDP (cf. Equation 3.19), only the following changes need to be applied. For the starting column  $i_s$ , the same formula is used.

$$\min_{w_{i_s,j}} \left\{ \sum_j [d_{i_s,j}(w_{i_s,j}) + T(w_{i_s,j-1}, w_{i_s,j}) + H_L(i_s, j, w_{i_s,j}) + H_R(i_s, j, w_{i_s,j})] \right\} \quad (4.1)$$

However for all other columns the optimization is altered in Equation 4.2 and Equation 4.3. Instead of using the branches in both directions, only the branches in one direction are used, since for the rest, we have an already fixed alignment from the previous columns. The warping penalty on the other hand now includes the horizontal predecessor.

$$\forall i^* > i_s : \min_{w_{i^*,j}} \left\{ \sum_j [d_{i^*,j}(w_{i^*,j}) + T(w_{i^*,j-1}, \tilde{w}_{i^*-1,j}, w_{i^*,j}) + H_R(i^*, j, w_{i^*,j})] \right\} \quad (4.2)$$

$$\forall i^* < i_s : \min_{w_{i^*,j}} \left\{ \sum_j [d_{i^*,j}(w_{i^*,j}) + T(w_{i^*,j-1}, \tilde{w}_{i^*+1,j}, w_{i^*,j}) + H_L(i^*, j, w_{i^*,j})] \right\} \quad (4.3)$$

Although with this modification of TSDP horizontal and vertical dependencies are considered, due to the hard decisions after each column no efficiency is lost. In detail, the complexity is composed of the following parts.

- $I \cdot J \cdot U \cdot V \cdot 9$  for the forward and backward run in the tree branch computation as with TSDP. In TSDP this complexity is counted twice, once for each run of dynamic programming. However in this case, not all forward and backward scores are needed. Only the forward score from 1 to  $i_s$  and the backward scores from  $I$  to  $i_s$  are used.

- $I \cdot J \cdot 9 \cdot 9$  for the alignment of the columns. For each column, each pixel in that column needs to be aligned. But since now a predecessor is given, there are only 9 candidates for each such pixel due to the Sakoe constraints. Additionally, there are 9 candidates for the previous vertical pixel.

Overall, this leads to a complexity of  $I \cdot J \cdot U \cdot V \cdot 9 + I \cdot J \cdot 9 \cdot 9$ , which is even less than for the TSDP. However, this approach is very heuristic. The alignment of the first column strongly influences the alignment of all other columns. Again the tree branches serve as look-ahead and look-before respectively, since they give us an estimate of the future that is not aligned yet.

## 4.2 Two-Level Dynamic Programming

The Two-Level Dynamic Programming (2LDP) is another warping method with the goal to maintain the horizontal and vertical first-order dependencies and enforce the Sakoe constraints. Similar to the previously introduced P2DW, the 2LDP divides the general warping criterion in Equation 3.2 into a global and a column level [Ney & Dreuw<sup>+</sup> 10].

$$\min_{\{w_{ij}\}} \left\{ \sum_{i,j} [d_{ij}(w_{ij}) + T(w_{i-1,j}, w_{i,j-1}, w_{ij})] \right\} \quad (4.4)$$

$$= \min_{\{w_{ij}\}} \left\{ \sum_i \min_{\substack{\{w_{ij}\}=\{w_j\} \\ \{w_{i-1,j}\}=\{\tilde{w}_j\}}} \left\{ \sum_j [d_{ij}(w_j) + T(\tilde{w}_j, w_{j-1}, w_j)] \right\} \right\} \quad (4.5)$$

For the inner minimization of the resulting leveled warping criterion in Equation 4.5 the column  $i$  is fixed and the optimization is done for all sequences  $\{w_j\}$  and the predecessor sequences  $\{\tilde{w}_j\}$  for the previous column  $i - 1$ . The notation  $w_j$  instead of  $w_{ij}$  emphasizes the fixation of  $i$ . The outer minimization is done over the columns. This division into two levels simplifies the computation of the warping mapping. On the other hand, the equality from Equation 4.4 to Equation 4.5 is not exact. Optimizing Equation 4.5 is therefore a relaxation of the general warping criterion.

As the minimization is done over the complete sequences  $\{w_j\}$  and  $\{\tilde{w}_j\}$ , the complexity is still exponential in  $J$  and therefore infeasible for practical applications. To overcome this high complexity, the key feature of the 2LDP is a restriction to a representative point  $w = w_{j^*}$  over which the optimization is performed. The representative is defined by a specific row  $j^*$  that is naturally set to  $J/2$  since this is expected to be the most important point during the optimization. However, also other values from 1 to  $J$  are possible. This idea is illustrated in Figure 4.2. Considering column  $i$  the representative is given by the black pixel  $(i, j^*)$  in the test image (cf. Figure 4.2(a)) and  $w_{ij^*}$  in the reference image (cf. Figure 4.2(b)). The mapping of column  $i$  is optimized given the possible predecessors. The result is a set of paths in the reference image, one for each possible representative  $w_{ij^*}$ . Since during the optimization

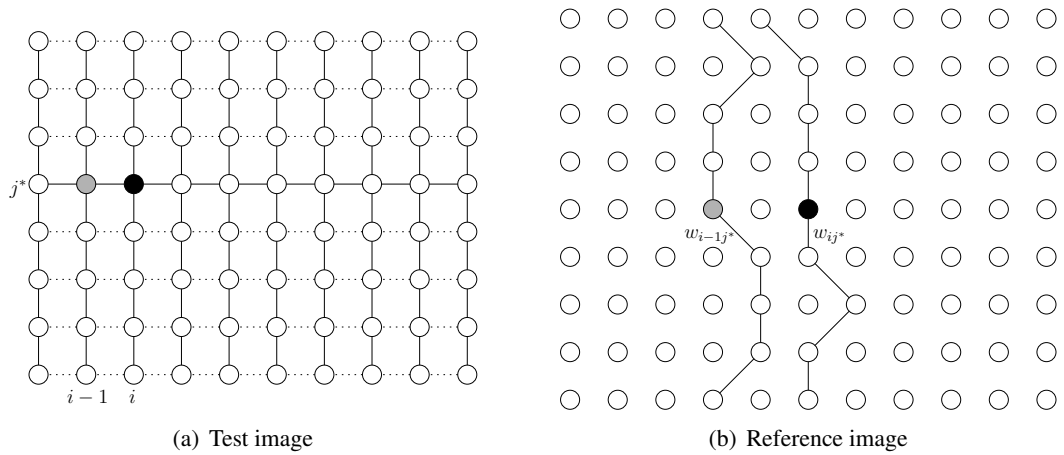


Figure 4.2: 2LDP: The optimization is done in two levels, a global and a column level. To avoid exponential complexity optimize over representative  $w_{ij^*}$ .

of a column the predecessors are already computed, the Sakoe constraints can be applied. The first-order dependencies are demonstrated in Figure 4.2(a). They are considered in both dimensions, however for the horizontal dependencies, the real optimization is only done for the representative row indicated by the dotted lines.

In the next section, the algorithm will be explained in more detail.

### 4.2.1 Dynamic Programming

To solve the optimization dynamic programming is used. First, the following notation is introduced. For a given column  $i$ ,  $w_1^J$  denotes the complete sequence of the mappings of all pixels in that column. Analogously  $\tilde{w}_1^J$  denotes the sequence for the predecessor column.

$$w_1^J = \{w_{i1}, \dots, w_{iJ}\} \quad (4.6)$$

$$\tilde{w}_1^J = \{w_{i-1,1}, \dots, w_{i-1,J}\} \quad (4.7)$$

The two-level optimization criterion (cf. Equation 4.5) can be solved with the following dynamic programming recursion [Ney & Dreuw<sup>+</sup> 10].

$$D(i, w_1^J) = \min_{\tilde{w}_1^J} \left\{ D(i-1, \tilde{w}_1^J) + d(i, \tilde{w}_1^J, w_1^J) \right\}, \quad (4.8)$$

$$d(i, \tilde{w}_1^J, w_1^J) = \sum_j [d_{ij}(w_j) + T(\tilde{w}_j, w_{j-1}, w_j)] \quad (4.9)$$

$D(i, w_1^J)$  represents the global level, while  $d(i, \tilde{w}_1^J, w_1^J)$  represents the column level. As can be seen in the global level, the minimization is done over the complete sequence  $w_1^J$ .

Assuming there are  $W$  options for each  $w_j$ , this leads to a complexity of  $I \cdot J \cdot W^{2J}$ . The global dynamic programming table has a complexity of  $I \cdot W^J$ , since for each column  $i$  there are  $W^J$  options for the sequence  $w_1^J$ . For each entry in this table the minimization has a complexity of  $J \cdot W^J$ , since it is done over all possible predecessors. In case the Sakoe constraints are applied, there are only nine possible predecessors for each pixel, but the complexity is still exponential in  $J$ . As already mentioned, this complexity is reduced by optimizing only over one representative. In detail, instead of considering the complete sequence  $w_1^J$ , only one  $w_{j^*}$  is selected for the minimization. In order to adapt the dynamic programming recursion accordingly, first a new auxiliary data structure needs to be defined [Ney & Dreuw<sup>+</sup> 10].

$$\hat{w}_1^J(i, w) := \arg \min_{w_1^J: w_{j^*}=w} \left\{ \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) + d(i, \hat{w}_1^J(i-1, \tilde{w}), w_1^J) \right\} \right\} \quad (4.10)$$

This data structure (cf. Equation 4.10) contains for each pair of column  $i$  and representative  $w$  the best path of mapping column  $i$  to the reference image, where the representative is mapped on  $w$ . This is achieved by computing the best path given the possible predecessors and storing the path rather than the score. Now the dynamic programming recursion can be altered [Ney & Dreuw<sup>+</sup> 10].

$$D(i, w) := \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) + \hat{d}(i, \tilde{w}, w) \right\} \quad (4.11)$$

$$= \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) + \min_{w_1^J: w_{j^*}=w} \left\{ d(i, \hat{w}_1^J(i-1, \tilde{w}), w_1^J) \right\} \right\} \quad (4.12)$$

The optimization is now done only over  $w$  leading to a much improved complexity. While the global dynamic programming table  $D(i, w)$  now has a complexity of  $I \cdot U \cdot V$ , for each entry a minimization over the predecessors of the representative has to be done, which has a complexity of  $W$  and nine for the Sakoe constraints. This is illustrated in Figure 4.3 that demonstrates the dynamic programming implementation of the global level. Considering entry  $(i, w)$  marked blue in the three-dimensional table, the Sakoe constraints allow nine predecessors marked in the gray. For the entry  $(i, w)$  the best path of mapping column  $i$  where  $(i, j^*)$  is mapped onto  $w$  is computed for each predecessor. Out of these nine paths, the best *path* is chosen and stored in the auxiliary data structure, while the corresponding best *score* global dynamic programming table.

In order to be able to trace back the decisions in the end, also the transitions from one column to the next are stored in a trace back table. In the final step, the score of the optimal warping can be found by minimizing over the entries in  $D$  for the last column.

$$score := \min_w \{D(I, w)\} \quad (4.13)$$

The warping itself can then be read from the auxiliary data structure for the single paths and the trace back table. The value of the minimal entry for the last column is also the overall cost

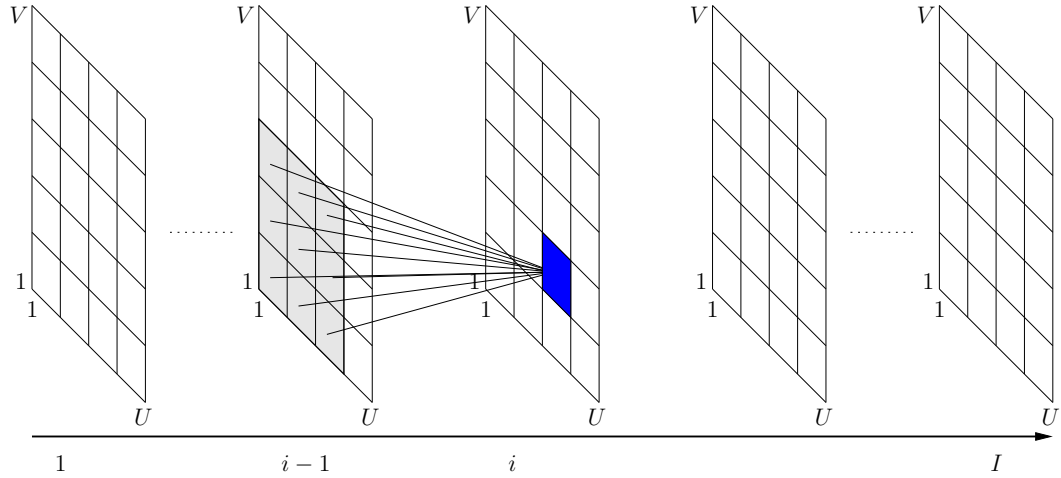


Figure 4.3: Global level of the 2LDP: Each entry  $(i, w)$  in the three-dimensional dynamic programming table corresponds to the best path of mapping  $i$  such that  $w_{ij^*} = w$ , given the nine predecessors.

of the final warping. The next two sections describe how the table  $D$  is filled in detail, starting with the initialization, where the entries  $D(1, w)$  are computed.

## 4.2.2 Initialization

During the initialization there are no predecessor paths. That means the best paths for all possible mappings of the first representative  $(1, j^*)$  have to be found. Since there are no restrictions other than the vertical Sakoe constraints on the first column ( $i = 1$ ) of the test image, every pixel of the reference image is such a possible mapping. This means, for all pixels  $(u, v)$  the best path where  $(1, j^*)$  is mapped onto  $(u, v)$  has to be computed. To do this in an efficient way, dynamic programming is used. First, a bottom-up run from 1 to  $j^*$  is performed, where for each  $j \in \{1, \dots, j^* - 1\}$  the best path through each pixel  $(u, v)$  is computed. The same is done for each  $j \in \{J, \dots, j^* + 1\}$  in a top-down fashion. As a result, we get two look-up tables that encode for each  $(u, v)$  the best path where  $j^* + 1$  and  $j^* - 1$  respectively are mapped onto  $(u, v)$ . The best paths for  $j^*$  can now easily be computed by connecting the two best paths. The process is illustrated in Figure 4.4. For the  $j^*$ -layer of the three-dimensional dynamic programming table, every pixel  $(u, v)$  is selected once as representative. The best path through this representative can be read from the next and previous layer respectively.

Overall, this leaves us with a complexity of  $9 \cdot U \cdot V \cdot (J - 1)$  for the look-up tables, since for each  $j$ -position except  $j^*$ , each  $(u, v)$  is considered and optimized over the 9 predecessors. Another  $9 \cdot U \cdot V$  is added for putting together the final paths, leading to a total of  $9 \cdot U \cdot V \cdot J$ .

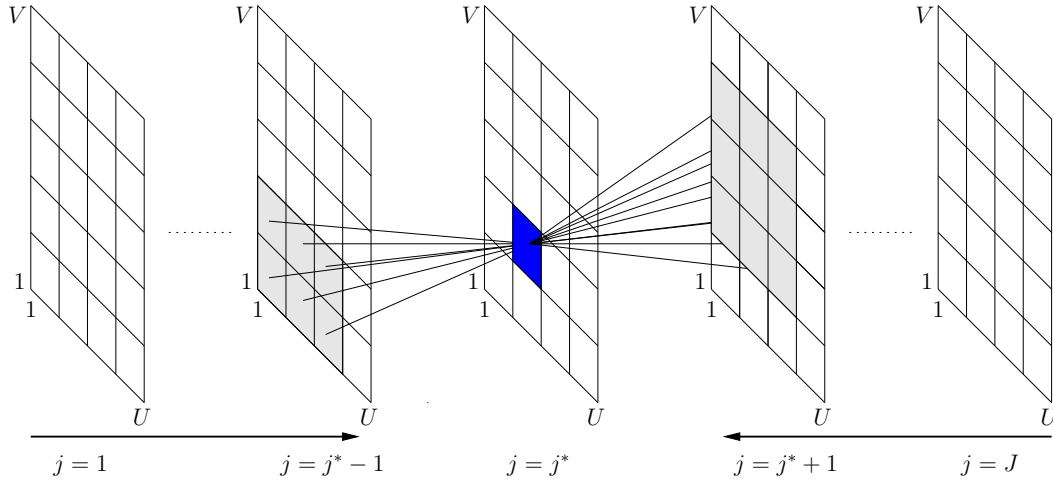


Figure 4.4: Initialization phase of the 2LDP: One bottom-up and one top-down run towards  $j^*$  create two look-up tables from which the best path is computed for each  $w_{j^*}$ .

### 4.2.3 Column-level

After the initialization, the global dynamic programming table  $D(i, w)$  is filled for all  $i > 1$ . As mentioned earlier, when computing the score for one entry  $(i, w)$  and its corresponding path, all nine predecessor paths have to be considered. Since the computation of the current path depends on the predecessor, nine paths have to be computed of which the best one is chosen. However, all neighboring entries regarding  $w$  share some of the predecessors. This fact can be exploited to avoid having to recompute all nine paths from scratch for each entry. Instead, for one predecessor, the dynamic programming tables are computed similar to the initialization phase, except that now the possibilities for each  $j$  position are greatly reduced due to the dependency on the previous path and the Sakoe constraints. This is illustrated in Figure 4.5. The predecessor path is given by the black connections. For each  $j$ , the Sakoe constraints allow for the next path to be positioned within a field of nine successors. The two look-up tables, one for the bottom-up and one for the top-down run are computed again until the representative is reached. Now both tables can be used to compute nine successor-paths at once. The complexity of one of such computations is given by  $9 \cdot 9 \cdot J$ , since for every  $j$ , there are nine possibilities and a minimization over the nine predecessors is performed.

### 4.2.4 Complexity

The overall complexity is now composed as follows.

- $9 \cdot U \cdot V \cdot J$  for the initialization.
- $81 \cdot J \cdot I \cdot U \cdot V$  for the column-level after the initialization.

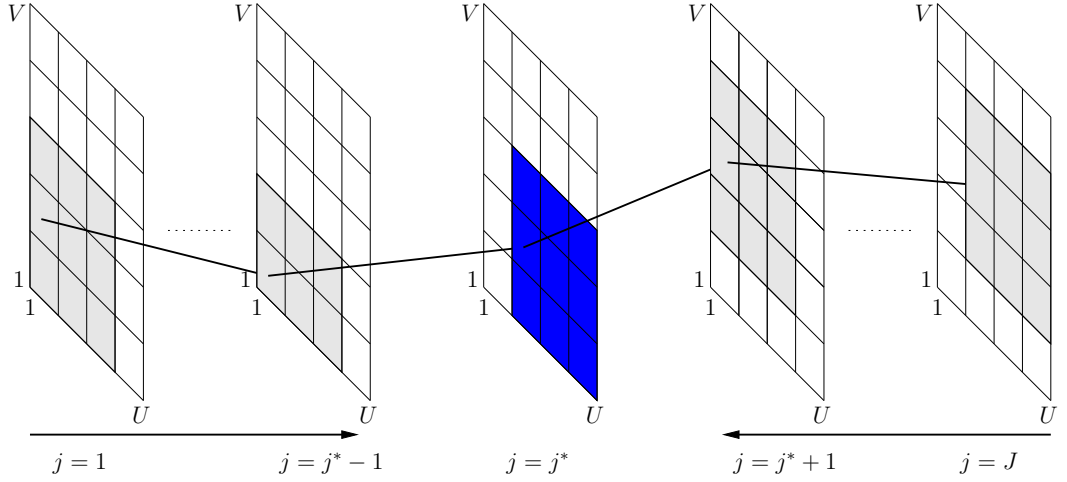


Figure 4.5: 2LDP on column-level: Similar to the initialization two look-up tables are computed, however now they are restricted by the predecessor path.

- $9 \cdot I \cdot U \cdot V$  for determining the best paths.

The result is a complexity of  $9 \cdot U \cdot V \cdot J + 81 \cdot J \cdot I \cdot U \cdot V + 9 \cdot I \cdot U \cdot V$ .

### 4.3 Restriction to $u$ -component

In the basic version of Two-Level Dynamic Programming, the optimization is performed over the complete  $w = (u, v)$ . One way to simplify this is to restrict the optimization in the global level only to the  $u$ -component of  $w$  (2LDP-U). On the one hand, this modification leads to a much improved complexity. On the other hand, it allows to draw conclusions about whether all the candidates considered by the basic 2LDP are actually necessary. To apply the modification, the Equation 4.10 and Equation 4.11 are altered in the following way.

$$\hat{u}_1^J(i, u) := \arg \min_{w_1^J: w_{j^*} = (u, \cdot)} \left\{ \min_{\tilde{u}} \left\{ D(i-1, \tilde{u}) + d(i, \hat{w}_1^J(i-1, \tilde{u}), w_1^J) \right\} \right\} \quad (4.14)$$

$$D(i, u) := \min_{\tilde{u}} \left\{ D(i-1, \tilde{u}) + \hat{d}(i, \tilde{u}, u) \right\} \quad (4.15)$$

$$= \min_{\tilde{u}} \left\{ D(i-1, \tilde{u}) + \min_{w_1^J: w_{j^*} = (u, \cdot)} \left\{ d(i, \hat{w}_1^J(i-1, \tilde{u}), w_1^J) \right\} \right\} \quad (4.16)$$

The  $w$  as representative and its predecessors  $\tilde{w}$  are replaced by simply  $u$  and  $\tilde{u}$  respectively. Note that when specifying the mapping of the representative in the auxiliary data structure to describe the best paths, the  $v$ -component is not specified. This means that  $v$  is still variable. Neighboring representatives can still be mapped to pixels with different  $v$ -components.

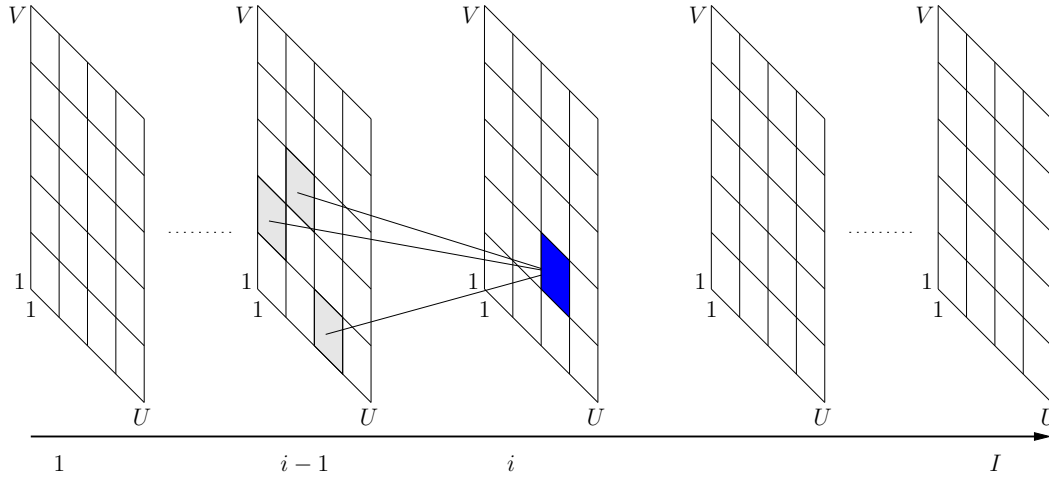


Figure 4.6: Global level of the 2LDP-U: The three-dimensional dynamic programming table is reduced to a two-dimensional table. However, each entry can have a different value for the  $v$ -component.

These variable options for  $v$  are just not included in the optimization. For each  $u$  option there is just one fixed option for  $v$ . Figure 4.6 demonstrates the alterations on the global level. Instead of nine predecessors, there are now only three predecessors to choose from. The three-dimensional table is reduced to a two-dimensional table.

The advantage of applying this simplification is a decrease of complexity. By discarding the  $v$ -component in the global optimization, the factor  $V$  disappears in the complexity for the global level. This leads to a global complexity of only  $3 \cdot I \cdot U$ .

As a disadvantage, the computed score is likely to be not as good as with the base version of the 2LDP, since we significantly reduce the number of candidates we consider in each step.

## 4.4 Local-Best

Another possibility to restrict the 2LDP is to apply the local-best selection strategy (2LDP-LB). When considering entry  $(i, w)$  in the global dynamic programming table of the 2LDP, nine paths are evaluated, one according to each predecessor. Each of these paths leads to a score, the path with the best score is selected. Note that these scores are composed of the score for the predecessor plus the additional costs for the new path. With the local-best strategy, only one predecessor and its path is considered, i.e. the one that has the best score without the costs of the new path. This is illustrated in Figure 4.7. From the nine possible predecessors, one has the best score and is therefore chosen immediately.

To apply the local-best strategy, the minimizations in the 2LDP-formulas (cf. Equation 4.10 and Equation 4.11) have to be changed. They are now done only over the predecessor entries

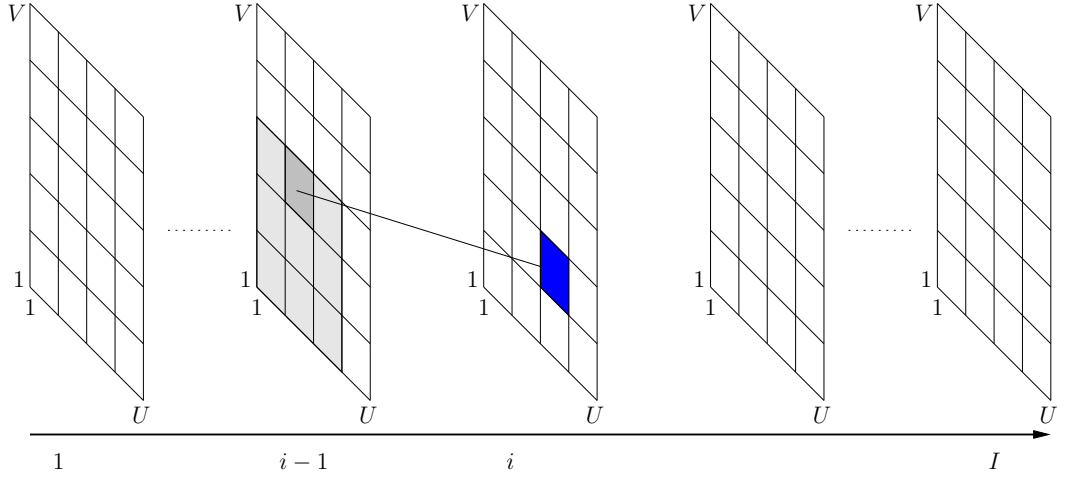


Figure 4.7: Global level of the 2LDP-LB: The predecessor with the best score is chosen immediately.

in the global table  $D$  without adding the column-score for the new path.

$$\hat{w}_1^J(i, w) := \arg \min_{w_1^J: w_{j^*} = w} \left\{ \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) \right\} + d(i, \hat{w}_1^J(i-1, \tilde{w}), w_1^J) \right\} \quad (4.17)$$

$$D(i, w) := \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) \right\} + \hat{d}(i, \tilde{w}, w) \quad (4.18)$$

$$= \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) \right\} + \min_{w_1^J: w_{j^*} = w} \left\{ d(i, \hat{w}_1^J(i-1, \tilde{w}), w_1^J) \right\} \quad (4.19)$$

The complexity of the global level remains unchanged, since still for each entry of the global dynamic programming table a minimization over the predecessors has to be done, even though the new paths are not considered anymore during that minimization. On the column level however, there is a slight difference. Let  $\Lambda$  be the number of different local best predecessors, in the worst case  $\Lambda = I \cdot U \cdot V$ . Since only the successor paths of the local best predecessors are needed, the complexity of  $81 \cdot J \cdot I \cdot U \cdot V$  for the column level after the initialization is reduced to  $81 \cdot J \cdot \Lambda$ . For the worst case the complexity is again the same as for the original 2LDP.

## 4.5 Strip-Restriction

While the Pseudo-2D warping model is extended by the First-Order Strip Extension, the same idea can be applied to the Two-Level Dynamic Programming, only that in this case it is rather a strip-restriction (2LDP-S) [Ney & Dreu<sup>+</sup> 10]. The 2LDP in general allows arbitrary paths through the reference image. However, in case it is known in advance that the objects to

be warped are relatively well aligned, one can restrict the paths to be placed within a strip defined by a strip-width around the representative. This restriction forces the paths to be oriented vertically and excludes paths with large horizontal deviations. To formalize this, for each path  $w_1^J$  the following restriction is enforced.

$$\forall j \in \{1, \dots, J\} : |u_{j^*} - u_j| \leq \Delta \quad (4.20)$$

For all  $j$ -positions, the  $u$ -component of  $w_j$  is allowed to differ by at most  $\Delta$  from the  $u$ -component of the representative. Thereby,  $\Delta$  defines the strip-width, i.e. the absolute width of the strip is given by  $(2\Delta + 1)$ . In the dynamic programming formulas, the optimization of the column level needs to be adapted, such that it is done only over those sequences that fulfill Equation 4.20.

$$D(i, w) := \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) + \hat{d}(i, \tilde{w}, w) \right\} \quad (4.21)$$

$$= \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) + \min_{\substack{w_1^J: w_{j^*} = w \\ |u_{j^*} - u_j| \leq \Delta}} \left\{ d(i, \hat{w}_1^J(i-1, \tilde{w}), w_1^J) \right\} \right\} \quad (4.22)$$

The effect of the strip-restriction is illustrated in Figure 4.8 for the initialization phase, where the impact is the strongest. The first column needs to be mapped to the reference image, where each pixel in the reference image is tried once as representative. For  $w_{1j^*}$  there are several possible paths (cf. Figure 4.8(b)), of which the cost minimal is chosen. The possible paths are only restricted by the vertical Sakoe constraints, apart from that they can have any form. If now a strip-restriction with  $\Delta = 1$  is applied, the set of possible paths is altered (cf. Figure 4.8(d)) and the paths are forced to exhibit a smaller horizontal deviation.

One disadvantage of the strip-restriction is an increase in complexity. This is due to the nature of the dynamic programming tables. Consider for example the initialization of 2LDP (cf. Section 4.2.2). All paths for the initializations are computed by generating two look-up tables of dimension  $J/2 \cdot U \cdot V$  running towards the representative. Each entry is optimized over nine predecessors and for all possible  $w_{1j^*}$  the corresponding paths are computed by selecting the optimal paths in reach of  $w_{1j^*}$  from the look-up tables. Unfortunately, this procedure is not possible anymore with the strip-restriction, since now each entry of the look-up tables needs to depend on the strip defined by the representatives. This means that the same look-up tables can not be used for each possibility  $w_{1j^*}$  anymore. To be precise, only those  $w_{1j^*}$  with the same  $v$  component can be represented by the same look-up tables. Therefore, the complexity of the initialization is increased by a factor of  $U$ . On the other hand, for each different  $u$  as value for the  $u$ -component of  $w_{1j^*}$ , not the full tables (size  $J/2 \cdot U \cdot V$ ) are needed. Instead, the  $U$  part can be replaced by the overall strip-width, leading to a size of  $J/2 \cdot (2\Delta + 1) \cdot V$ . In summary, the overall complexity for the initialization is  $J \cdot (2\Delta + 1) \cdot V \cdot U$ . The same considerations can be done for the column-level following the initialization.

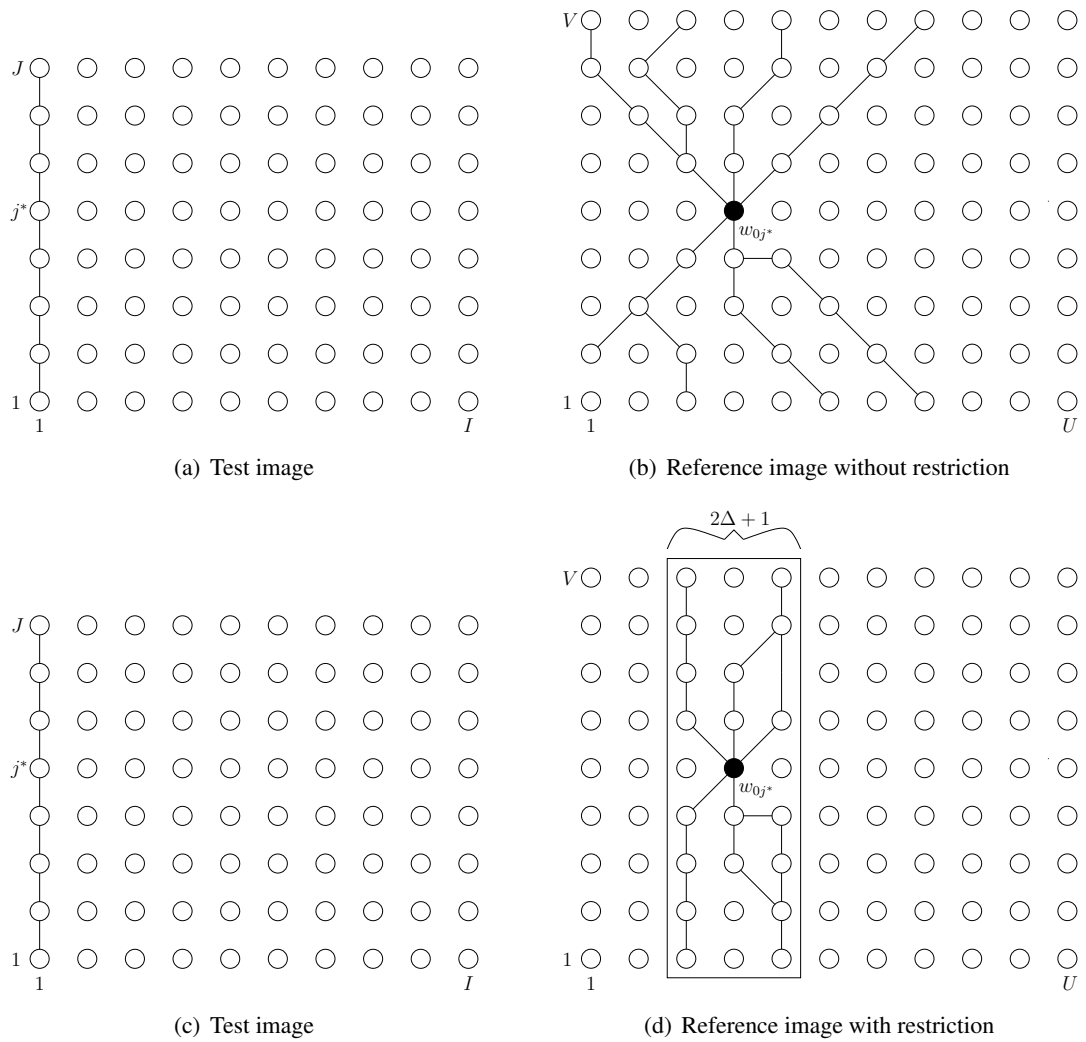


Figure 4.8: 2LDP-S in the initialization phase: In 2LDP the alignment of the first column of the test image (a) does have several options that are only restricted by the Sakoe constraints (b). With the strip-restriction, the horizontal deviation of the alignments for the first test column (c) is restricted by a strip-width (d).

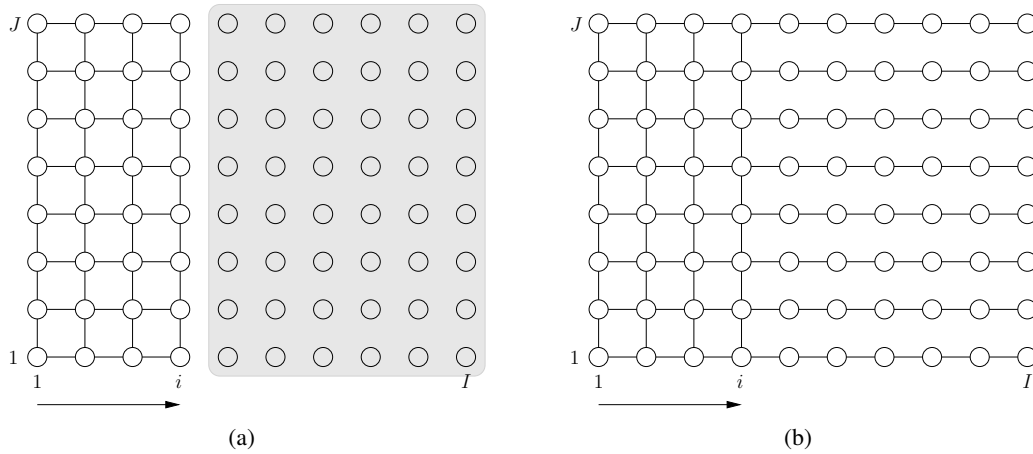


Figure 4.9: Lookahead: Use branches of a backward run of dynamic programming to get an estimate of the rest of the image.

## 4.6 Lookahead

Considering the basic version of 2LDP, the warping is computed column-wise. For each column  $i$  several candidates depending on their representatives are generated. During the generation of one such candidate, the nine predecessors are considered. For these nine predecessors, all columns from 1 to  $i - 1$  are already aligned and fixed. The alignment of the current column  $i$  is therefore influenced by all previous columns. However, the rest of the image, i.e. all columns from  $i + 1$  to  $I$  have no direct influence on the alignment of column  $i$ . Only during the candidate selection at the end of the optimization procedure there is an influence of the alignments of the following columns onto  $i$ . As a result, the alignment of column  $i$  is optimized only regarding the previous columns. This might be too greedy and locally good decision can lead to globally bad results in the end. This is problematic, since due to the Sakoe constraints it takes a long time to compensate a bad decision.

A solution to this problem is inspired by TSDP (cf. Section 3.5). When a column is aligned with TSDP, there is no information about the other columns available. Therefore, a forward and a backward run are used to get an estimate on the costs for the rest of the image. Something similar can be done for 2LDP. Since the warpings for the previous columns are already given, a backward run from the last column to the first can be used as lookahead (LA) to align the columns given the estimated costs of the following columns. This is demonstrated in Figure 4.9. During the alignment of column  $i$ , the rest of the warping (gray area, Figure 4.9(a)) is unknown. Therefore the branches computed by the backward run from  $I$  to  $i$  are used as lookahead (cf. Figure 4.9(b)).

The new lookahead is defined as given in Equation 4.23 and equals to the formula for the backward run performed during TSDP (cf. Equation 3.18). For any  $j$ -position in the current

column  $i^*$ , the lookahead corresponds to the mapping of the rest of this particular row that has the lowest costs given the distances and penalties.

$$D_{LA}(i^*, j, w_{i^*j}) := \min_{\{w_{ij}, i > i^*\}} \left\{ \sum_{i > i^*} [d_{ij}(w_{ij}) + T(w_{i-1,j}, w_{ij})] \right\} \quad (4.23)$$

The next step is to adapt the dynamic programming formulas of the 2LDP (cf. Equation 4.11) to include the lookahead. First of all, the lookahead is included when the paths are computed. The auxiliary data structure storing the paths therefore includes a term for the lookahead. The cost of the path for column  $i$  going through  $w$  as representative is extended by the cost of the lookahead-scores for each  $j$ -position.

$$\begin{aligned} \hat{w}_1^J(i, w) := & \arg \min_{w_1^J: w_{j^*} = w} \left\{ \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) + d(i, \hat{w}_1^J(i-1, \tilde{w}), w_1^J) \right. \right. \\ & \left. \left. + \gamma \cdot \sum_j D_{LA}(i, j, w_{ij}) \right\} \right\} \end{aligned} \quad (4.24)$$

The three terms in Equation 4.24 each account for one part of the image.  $D(i-1, \tilde{w})$  holds the costs for everything up to column  $i$ ,  $d(i, \hat{w}_1^J(i-1, \tilde{w}), w_1^J)$  the costs for column  $i$  itself and  $\sum_j D_{LA}(i, j, w_{ij})$  the costs for the following part of the image. The influence of the lookahead can be regulated by the parameter  $\gamma$ . The effect of this auxiliary data structure is that the paths are now aligned using the lookahead scores. However, these new costs should also be used in the candidate selections on the global level. Thus the dynamic programming formulas (cf. Equation 4.11) are altered again.

$$D(i, w) = \min_{\tilde{w}} \left\{ D(i-1, \tilde{w}) + d(i, \hat{w}_1^J(i-1, \tilde{w}), \hat{w}_1^J(i, w)) \right\} \quad (4.25)$$

When the decision for entry  $(i, w)$  is made, the minimization is again done over the predecessors. For that purpose, now the costs of the predecessor stored in the global dynamic programming table  $D$  plus the costs of the new path, computed with the lookahead scores, given the predecessor are considered. The lookahead scores are only considered during the minimization, they are not added to the entry  $(i, w)$  of  $D$ . As a result the entries in  $D$  still only capture the costs of the exact calculations so far. Therefore, the final result can still be obtained by minimizing over the entries of the last column  $I$ .

The complexity of calculating the lookahead is given by  $9 \cdot I \cdot J \cdot U \cdot V$  for the one backward run of dynamic programming. For the rest of the procedure, the theoretical complexity remains unchanged. However, in practice the complexity will be slightly increased since a few more operations are necessary to include the lookahead in the calculations.

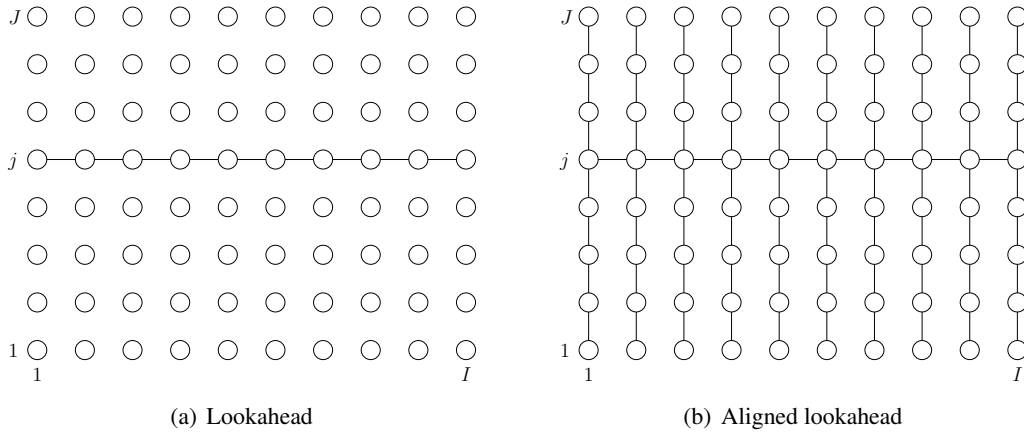


Figure 4.10: Aligned lookahead: Instead of optimizing only the row (a), optimize a tree where the branches are the columns of the image (b).

## 4.7 Aligned Lookahead

The lookahead defined in the previous section is based on only one backward run of dynamic programming. During this backward run, the best alignments of the rows is computed, independent from the other rows. As a result, when the lookahead at position  $(i, j)$  is applied, it is defined only by the rest of the specific row  $j$  and the best possibility to map this row into the reference image. The rest of the image has no influence leading to a very greedy lookahead. To improve this, the computation of the lookahead can be altered, such that the rows are not optimized independently. Before the backward run a bottom-up and top-down run of dynamic programming is performed. This way, the best alignments of the columns are determined in both directions. The results of the bottom-up and top-down run can then be used during the backward run to align the rows. This is illustrated in Figure 4.10. Again, trees are optimized, however this time they are rotated. The stem is defined by rows, the branches by columns (cf. Figure 4.10(b)). When this aligned lookahead (ALA) is applied at pixel  $(i, j)$ , it is not only defined by the rest of the row but by the complete rest of the image.

The details for the new lookahead will be given now. The formulas for the bottom-up and top-down run are very similar to the formulas for the forward and backward run (cf. Equation 3.17 and Equation 3.18).

$$D_{\text{Bottomup}}(i, j^*, w_{ij^*}) = \min_{\{w_{ij}, j < j^*\}} \left\{ \sum_{j < j^*} [d_{ij}(w_{ij}) + T(w_{i,j+1}, w_{ij})] \right\} \quad (4.26)$$

$$D_{\text{Topdown}}(i, j^*, w_{ij^*}) = \min_{\{w_{ij}, j > j^*\}} \left\{ \sum_{j > j^*} [d_{ij}(w_{ij}) + T(w_{i,j-1}, w_{ij})] \right\} \quad (4.27)$$

In order to use the top-down and bottom-up run in the backward calculations, an auxiliary

formula is defined in Equation 4.28. The sequence  $w_{i^*}^I$ , i.e. the mappings of all pixels in the same row, starting at position  $i^*$ , is calculated by summing up the distances, the penalties and the top-down and bottom-up scores. The sequence leading to the minimal costs for row  $j$  starting from position  $i^*$  is stored in  $\hat{w}_{i^*}^I(j)$ .

$$\hat{w}_{i^*}^I(j) := \arg \min_{\{w_{ij}, i > i^*\}} \left\{ \sum_{i > i^*} [d_{ij}(w_{ij}) + T(w_{i-1,j}, w_{ij}) + D_{\text{Topdown}}(i, j, w_{ij}) + D_{\text{Bottomup}}(i, j, w_{ij})] \right\} \quad (4.28)$$

With the help of the new auxiliary formula, the final lookahead can now be calculated by summing up the distances and penalties for the previously found sequence (cf. Equation 4.29). By this procedure, the new lookahead is only aligned by the bottom-up and top-down run, the additional scores are not used in the lookahead itself.

$$D_{ALA}(i^*, j, w_{i^*j}) := \min_{w_{i^*+1,j}} \left\{ T(w_{i^*}, w_{i^*+1}) + \sum_{i > i^*} [d_{ij}(\hat{w}_i(j)) + T(\hat{w}_i(j), \hat{w}_{i+1}(j))] \right\} \quad (4.29)$$

The minimization in Equation 4.29 is necessary, since when the lookahead is read at entry  $(i^*, j, w_{i^*j})$ , the vertical branches should only influence the alignments of pixels next to  $(i^*, j)$ . The aligned lookahead is then used in the dynamic programming procedure of the 2LDP the same way, as the first variant of the lookahead (cf. Equation 4.25). Since two additional dynamic programming runs are necessary, the complexity of the algorithm is increased. Both runs and the final minimization in Equation 4.29 add each the complexity of  $9 \cdot I \cdot J \cdot U \cdot V$ , leading to a total complexity of  $36 \cdot I \cdot J \cdot U \cdot V$  to calculate the aligned lookahead, in comparison to  $9 \cdot I \cdot J \cdot U \cdot V$  for the unaligned case.

## 4.8 Comparison of the Warping Algorithms

In the previous sections a number of warping algorithms have been introduced. In this section, the connections and similarities of the algorithms are investigated. The most obvious connection is between TSDP and C-TSDP, since the latter is an extension of the former. However the optimization of the columns is done independently with TSDP and as a result the order in which the columns are optimized does not matter. This is different for C-TSDP, where the columns have to be optimized ordered from the starting column to the edges of the image. Because of this restriction the scores achieved by TSDP are usually lower than the scores C-TSDP reaches. Yet there is no guarantee that the TSDP scores are a tight lower bound for C-TSDP, since TSDP is always using the forward and backward scores for alignment, while in C-TSDP, one of the two is replaced by the exact alignment so far.

If for C-TSDP the first column is chosen as starting column, it can be seen as a baseline for 2LDP combined with the basic lookahead (2LDP-LA). In this case, C-TSDP first aligns the first column given the backward scores, since for the first column, there are no forward scores. Then the next columns are aligned given the backward scores and the alignment of the previous column. 2LDP-LA does the same, except that not only one candidate for each column is considered. Instead for each column and each representative there is one candidate of which the best ones are chosen in the end. So in a way, 2LDP-LA extends C-TSDP by the global optimization level. It is important to note that given the same images  $X$  and  $R$ , the alignments computed by C-TSDP do not necessarily need to be also candidates during the optimization of 2LDP-LA. This is only the case for the first column. Let  $j^*$  be the representative row for 2LDP-LA and let  $w_1^j$  be the sequence found by C-TSDP where  $w_{j^*} = (u, v)$ . Then this is the best possible alignment for the first column given the backward scores. Since in the initialization of 2LDP-LA the best paths for all pixels including  $(u, v)$  as representatives are computed,  $w_1^j$  has to be one of them. However, already for the second column things might be different. The alignment found by C-TSDP has  $w_1^j$  as a fixed predecessor. 2LDP-LA on the other hand, has several options as predecessor and there is no guarantee that the same predecessor is chosen. This means, if for one column, the predecessor that C-TSDP is forced to use is not used by 2LDP-LA, the final warping of C-TSDP is not among the candidates for 2LDP-LA. In fact it is even possible that C-TSDP finds a warping with less costs than 2LDP-LA.

There is also an obvious similarity between 2LDP in general and P2DW, since both are divided into a global and a column level. 2LDP can be reduced to P2DW-FOSE by using the variant that is optimized only over the  $u$ -component (2LDP-U, Section 4.3) together with the strip-restriction with the same strip-width as P2DW-FOSE. Additionally the horizontal dependencies have to be discarded. The latter points out the most important difference between the two methods. P2DW optimizes the columns independently making it impossible to enforce the Sakoe constraints.

Although the approach of CTRW-S differs from the other warping models by being an approximation rather than a relaxation, also this model shows some similarities to the other models. In each iteration, the algorithm passes messages through the nodes once forward and once backward. During each pass, the information from the previous pass is used. This can be seen as a lookahead, similar to 2LDP lookahead variants or TSDP and C-TSDP.

### 4.8.1 Search Criterion

The different models that were introduced lead to different search criteria that need to be considered, when the methods, and especially the scores they produce, are compared. CTRW-S, C-TSDP and the 2LDP variants all search for warpings with Sakoe constraints. The other methods relax this criterion. P2DW-FOSE and TSDP search for warpings where the vertical Sakoe constraints are enforced, while ZOW uses no Sakoe constraints at all. The last three methods can therefore easier compute warpings with lower scores. On the other hand, these

Table 4.1: Sakoe constraint enforcement of the different warping algorithms.

Method	Vertical Sakoe constraints	Horizontal Sakoe constraints
ZOW	no	no
P2DW-FOSE	yes	no
TSDP	yes	no
CTRW-S	yes	yes
C-TSDP	yes	yes
2LDP	yes	yes

lower scores are at the expense of discarding some of the geometric structure of the object. Table 4.1 contains an overview over which part of the Sakoe constraints is enforced in which algorithm.

#### 4.8.2 Complexity

One important feature of the algorithms is the complexity. Table 4.2 gives an overview of the complexities of the different warping methods investigated. Taking 2LDP as baseline, the factor column of this table shows by which factor the complexity differs from the complexity of 2LDP. This factor however is just an approximation giving an idea how the complexities relate.

It is obvious that ZOW is by far the fastest of all methods, even if the warprange is unlimited. For a low strip-width also P2DW-FOSE is more efficient than 2LDP. However, choosing  $\Delta = 5$  already leads to a more complex method. We will see later that for optimal recognition performance, even higher strip-widths are needed. CTRW-S becomes more costly with a number of iterations higher than 5. TSDP needs roughly one third of the time 2LDP needs to compute the warpings. As already mentioned in Section 4.1, C-TSDP is slightly faster than TSDP.

While 2LDP-S extends the complexity by a factor proportional to the strip-width, 2LDP-U significantly reduces the complexity making it the second fastest method next to ZOW. The lookaheads are not methods themselves, their complexity therefore needs to be added to the variant of 2LDP they are applied to. Combining it with one of the 2LDP variants will therefore always have a greater complexity than TSDP. However, combined with the basic lookahead, 2LDP-U is still the second fastest method.

Table 4.2: Complexities of the warping methods. The factors are estimated assuming  $64 \times 64$  images.

Method	Complexity	Factor
ZOW	$I \cdot J \cdot \Delta^2$	$3 \cdot 10^{-6} \cdot \Delta^2$
P2DW-FOSE	$3 \cdot U \cdot I \cdot (1 + 3 \cdot (2\Delta + 1) \cdot J \cdot V)$	$1/9 \cdot (2\Delta + 1)$
TSDP	$I \cdot J \cdot U \cdot V \cdot 27$	$1/3$
CTRW-S	$N \cdot 2 \cdot I \cdot J \cdot 9 \cdot U \cdot V$	$2/9 \cdot N$
C-TSDP	$I \cdot J \cdot U \cdot V \cdot 9 + I \cdot J \cdot 9 \cdot 9$	$1/10$
2LDP	$9 \cdot U \cdot V \cdot J + 81 \cdot J \cdot I \cdot U \cdot V + 9 \cdot I \cdot U \cdot V$	$1$
2LDP-U	$9 \cdot U \cdot V \cdot J + 81 \cdot J \cdot I \cdot U + 3 \cdot I \cdot U$	$1/64$
2LDP-S	$(2\Delta + 1) \cdot (9 \cdot U \cdot V \cdot J + 81 \cdot J \cdot I \cdot U \cdot V) + 9 \cdot I \cdot U \cdot V$	$1 \cdot (2\Delta + 1)$
2LDP-LB	$9 \cdot U \cdot V \cdot J + 81 \cdot J \cdot \Lambda + 9 \cdot I \cdot U \cdot V$	$1^*$
LA	$I \cdot J \cdot U \cdot V \cdot 9$	$1/9$
ALA	$I \cdot J \cdot U \cdot V \cdot 36$	$4/9$

\*Worst case of  $\Lambda = I \cdot U \cdot V$  assumed.



# Chapter 5

## Practical Aspects

### 5.1 Distance Caching

All warping algorithms introduced in the previous chapters need to access the distances  $d_{ij}(w_{ij})$  between the pixels and their mappings. In the complexity analysis for the algorithms, this access was assumed to have a complexity of 1. On the other hand, as we have seen in Chapter 2 to compute the distance a distance function such as the absolute distance has to be evaluated. Since this is done on feature level, this operation is quite expensive to be done several times. E.g. given a 30-dimensional feature vector there are 30 subtractions and 29 additions necessary to compute the absolute distance of two pixels. For this reason the distances are computed only once and then saved in a look-up table, the distance cache. In general, the distance from each pixel in the test image to each pixel in the reference image is accessed at least once. Therefore the distance cache has a complexity of  $I \cdot J \cdot U \cdot V$ .

The same is done for the penalties. However, since only skips of at most one need to be considered, the penalty cache has a constant complexity.

### 5.2 Distance Thresholding

One problem with using pixel distances is that a single pixel can have an unproportionally large influence on the distortions. This is also the case for pixels that do not belong to the objects that are supposed to be warped such as background noise or occlusions. Especially the latter is problematic, since occlusions occur within the object and do not simply influence the final score, but also the warping of the neighboring pixels due to the dependencies. For this reason, distance thresholding is used [Gass & Pishchulin<sup>+</sup> 11]. By introducing a maximum  $\tau$  for all distances, the influence of a single pixel is restrained.

$$\hat{d}_{ij}(w_{ij}) = \min\{\tau, d_{ij}(w_{ij})\} \quad (5.1)$$

To apply the distance thresholding, the old distance function  $d_{ij}(w_{ij})$  is replaced by the function  $\hat{d}_{ij}(w_{ij})$  in the formulas introduced in the previous chapters.

### 5.3 Nearest Neighbor Pruning

When the nearest neighbor rule is used for classification based on similarity scores, the exact scores themselves are not needed. The decision only depends on which test object leads to the lowest score and not what that score actually is. This can be exploited to do pruning, when the computation of the scores is done sequential [Gass & Pishchulin<sup>+</sup> 11]. Since the goal is to find the reference image leading to the minimal score, the current minimum is kept in memory. As soon as it becomes evident that the next warping that is computed will lead to a higher score, the computation can be canceled. In order to detect this, lower bounds are needed for the algorithms. For the CTRW-S, the lower bound computed from the solution of the subproblems (cf. Section 3.6) can be used [Gass & Pishchulin<sup>+</sup> 11]. As described in [Gass & Pishchulin<sup>+</sup> 11], a simple lower bound can be found for the other warping methods by summing up the minimal distances for each pixel if warped to any arbitrary pixel in the reference image. The lower bound is given by the following formula.

$$lb = \sum_{ij} \min_{w_{ij}} \{d_{ij}(w_{ij})\} \quad (5.2)$$

This can be computed on the fly during the distance caching. However, the lower bound obtained by the method is very weak. Therefore, this is extended in this thesis as follows. Each method introduced above optimizes the warpings column-wise. That means, after each column a partial result is available. This can be used to update the lower bound. Let  $D(i, \Theta)$  be the global dynamic programming table of the method that is used, e.g.  $D(i, w)$  for the 2LDP or simply  $D(i)$  for the TSDP. The new lower bounds are computed by the following formula.

$$lb_{i^*} = \min_{\Theta} \{D(i^*, \Theta)\} + \sum_{i > i^*, j} \min_{w_{ij}} \{d_{ij}(w_{ij})\} \quad (5.3)$$

By using this method, it is guaranteed that the lower bound can only increase, since the dynamic programming tables can not contain scores lower than the absolute minimum warpings of independent pixels. It is also still a tight lower bound, since for each column the minimal possible score computed by the optimization algorithm is taken. With each optimized column, the lower bound gets closer to the minimal distance computed so far and the optimization can be cancelled earlier, when the lower bound exceeds the minimal distance.

# Chapter 6

## Databases

In this chapter, two databases that are used for the evaluation are introduced. The first database is a modified subset of the AR-Face database [Martinez & Benavente 98] and the second database is the pose subset of the CMU-PIE database [Sim & Baker<sup>+</sup> 02].

### 6.1 Rotated-45 AR-Face

The AR-Face database [Martinez & Benavente 98] in general offers face images with different facial expressions, illuminations and occlusion. Additionally, each image was recorded twice in two sessions two weeks apart adding effects caused by time differences, such as different hair styles.

The Rotated-45 AR-Face database is composed of a subset capturing 30 classes of the original database. The reduction to 30 classes is done for efficiency reasons. For testing seven images containing different facial expressions and illuminations from the second recording session are used. For training, the corresponding seven images from the first session are used. This leads to a total of 210 reference and 210 test images. To raise the level of difficulty, the test images were edited by artificial rotations. The degree of rotation for each image is a randomly selected value between -45 and 45 resulting in a database with rotations of up to 45 degrees clock-wise and counter clock-wise. After the images are rotated they are resized to  $64 \times 64$  images. One effect of the rotation and resizing are different scalings for the images that also add to the difficulty level. An example for one class is given in Figure 6.1.

### 6.2 CMU-PIE

The CMU Pose, Illumination and Expression database (CMU-PIE) [Sim & Baker<sup>+</sup> 02] offers face images with different facial expressions, illuminations and poses. In this thesis, the evaluation is done on the pose subset of the database. It consists of images for 13 different poses of 68 classes. The poses vary from profile to frontal images (cf. Figure 6.2). The original images are cropped to contain only the faces. The used setup is the so called mug-shot training, where only a single reference image per class is used. For this purpose the frontal image is selected, all other poses are used for testing. This results in 68 training images and 816 test images.

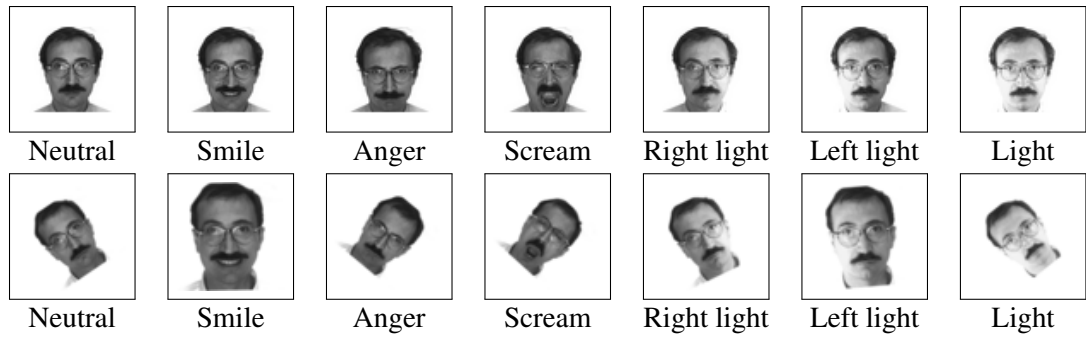


Figure 6.1: Rotated-45 AR-Face database: The test images in the second row are rotated by random degrees between  $-45$  and  $45$ .

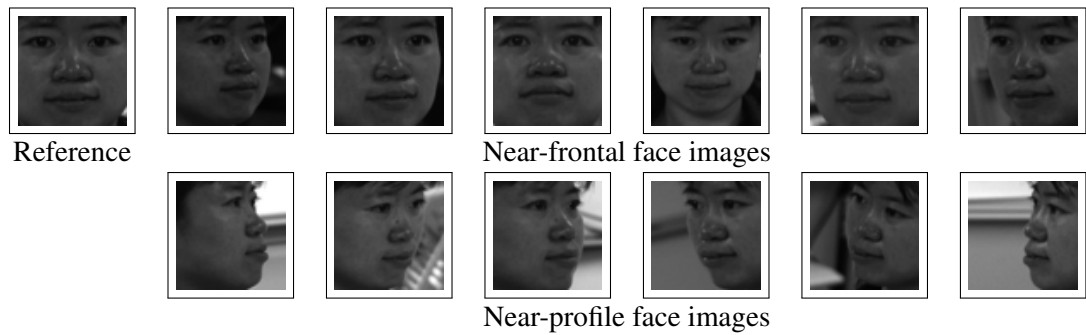


Figure 6.2: CMU-PIE pose database: One reference image and twelve test images per class. The test images are clustered into near-frontal and near-profile images.

As illustrated in Figure 6.2, the test images can be clustered into near-frontal and near-profile images. Similar to the AR-Face database the images are resized to  $64 \times 64$ .

# Chapter 7

## Results

In this chapter the results for the developed warping algorithms (cf. Chapter 3 and Chapter 4) will be presented and compared based on error rates and the warping scores. The evaluation is done on the two databases from Chapter 6 with an emphasis on the Rotated-45 AR-Face database. For the CMU-PIE database the same setup as in [Gass & Pishchulin<sup>+</sup> 11] is used. That means the warping is reversed such that the reference images are mapped onto the test images and for the comparison with the near-profile images only the corresponding half of the reference image is used. Since the images are vertically well aligned, the PCA-U-SIFT feature descriptors and local context is used for computing pixel distances (cf. Chapter 2). On the Rotated-45 AR-Face on the other hand, PCA-SIFT features and no local context is used to be invariant to rotations. For comparability the same penalties are used in all experiments, except for ZOW (cf. Section 7.1). Also the distance threshold  $\tau$  is kept constant for each database. For the runtime measurement a computer with a 2.2 GHz CPU and 16 GB RAM was used.

The evaluation is structured as follows. For each method introduced in Chapter 3 and Chapter 4, first the recognition performance on both databases is analyzed. This is followed by a qualitative evaluation to illustrate the characteristics of the algorithms. For this purpose three examples from the Rotated-45 AR-Face database. The first example is for a test image with no rotation, but scaling, the second example is for an image with medium rotation of 25 degrees, and the last example shows an image with the maximum rotation of 45 degrees. For each example the test image  $X_{ij}$ , the reference image  $R_{uv}$ , the distorted reference image  $R_{w_{ij}}$  and the distortion grid for  $w_{ij}$  are included. Since the warping mapping was defined by mapping the test pixel to the reference pixels (cf. Equation 3.1), it is the reference image that is distorted. The distorted image can be obtained by going through the grid of the test image selecting the image values of  $w_{i,j} = (u, v)$  for each  $(i, j)$ , leading to  $R_{w_{ij}}$  as the distorted reference image. One has to keep in mind that the warpings  $w_{ij}$  were not computed using the gray levels shown in the example. Also for the examples 30-dimensional PCA-SIFT feature vectors were used during the computation.

On score-level, the comparison makes only sense for methods that implement the same search criterion (cf. Table 4.1). Models with different search criterion apply the penalty in different ways, e.g. the P2DW-FOSE applies the horizontal penalty only based on the center of the strip. The Sakoe consistent methods on the other hand apply the penalties pixel-wise. The independence in one or both dimension also makes it easier to achieve lower scores due

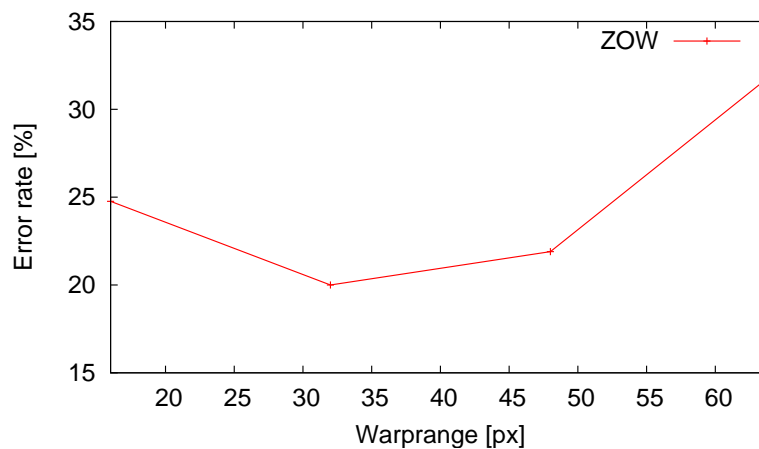


Figure 7.1: Rotated-45 AR-Face: Results for ZOW with different warpranges  $\Delta$ .

to less restrictions.

The next sections are organized as follows. In Sections 7.1 to 7.4 the state-of-the-art warping algorithms introduced in Chapter 3 will be discussed. In Section 7.5 the performance of Constrained Tree-Serial Dynamic Programming (cf. Section 4.1) is analyzed. Section 7.6 follows with the analysis of Two-Level Dynamic Programming (cf. Section 4.2). The positive effect of the lookahead will be shown and algorithm will be compared to the various modifications introduced in Section 4.2. Finally, in Section 7.8 a summary of the recognition results on both databases will be given.

## 7.1 Zero-Order Warping

The first model investigated is Zero-Order Warping. As the simplest available model, it provides a baseline for the experiments. The plot in Figure 7.1 shows the error rate for the recognition on the Rotated-45 AR-Face database given different warpranges. The best results are achieved using a medium warprange of  $\Delta = 32$ . Choosing a smaller warprange limits the warpings significantly and makes the compensation of the rotations difficult. Higher warpranges on the other hand also lead to worse results. This happens, since besides the penalty, the warprange is the only way for the ZOW to orientate the warping on the original structure of the image. The higher the warprange, the more structure can be destroyed during the warping. The medium warprange is therefore a good trade-off.

As mentioned in Section 3.3, the warping penalty for ZOW is an absolute penalty. The displacement of a pixel is penalized depending on the absolute distance between the pixel and its warping mapping. While this still works for well aligned objects, it is not suited for the Rotated-45 AR-Face database. Due to the rotations and scalings, the absolute penalties

Table 7.1: Rotated-45 AR-Face: Effect of absolute warping penalty.

Method	Penalty	Error rate [%]	Abs. errors
ZOW, $\Delta = 32$	-	20.00	42
ZOW, $\Delta = 32$	Euclidean distance	22.38	47

handicap the warping of the objects. This is underlined by the results in Table 7.1. By using the Euclidean distance as absolute penalty, the result has deteriorated.

As reported in [Gass & Pishchulin<sup>+</sup> 11], on the CMU-PIE database the ZOW performs well on the near-frontal face images with an error rate of 0.49%. However, on the near-profile images the error rate is at 31.61% leading to an average error rate of 16.05%.

The images for the three examples in Figure 7.2 underline the drawbacks of ZOW. There is no structure visible in the warping grid, meaning the original structure of the image is not maintained and the unrestricted nature for the ZOW is evident. The large warprange of  $\Delta = 32$  allows the model to pick for each pixel the best mapping within at least half the image, leading to the strong distortions visible in the grids and deformed reference images with many artifacts. However, the general structure of the test image is recreated for the scalings as well as the rotations.

## 7.2 Pseudo Two-Dimensional Warping

As already mentioned in Section 3.4 only the P2DW with the First-Order Strip Extension (P2DW-FOSE) is considered here. The plot in Figure 7.3 shows the error rates on the Rotated-45 AR-Face database achieved by the P2DW-FOSE using different strip widths. The best result of 15.71% was obtained by using  $\Delta = 15$ . This means that in absolute numbers the strip has a width of  $2\Delta + 1 = 31$  pixels. Given the  $64 \times 64$  images, this is already half the image. The recognition result is therefore at the expense of a high complexity. Choosing lower values for  $\Delta$  results in steadily increasing error rates. On the other hand, also higher values do not lead to a better recognition rate indicating that no better warpings can be found.

On the CMU-PIE pose database, the P2DW-FOSE achieves an error rate of 0.25% on the near-frontal and 17.63% on the near profile images [Pishchulin & Gass<sup>+</sup> 11]. This makes an average of 8.94%. For the experiments a value of  $\Delta = 3$  was used for the strip-width. The good alignment of the images leads to a good performance on near-frontal images. Since this is not given anymore for the near-profile images, the result deteriorates.

Figure 7.4 contains the three examples for warpings calculated by the P2DW-FOSE. The value for  $\Delta$  used for these images is 15 which leads to the best result achieved by the P2DW-FOSE on the Rotated-45 AR-Face database. Starting with the first example without rotation, it can be observed that the center of the distortion grid is maintained and the original 2D grid structure is still visible. Only at the borders of the grid distortions take place to compensate the scaling of the image. Due to the independent alignment of each column, the paths diverge

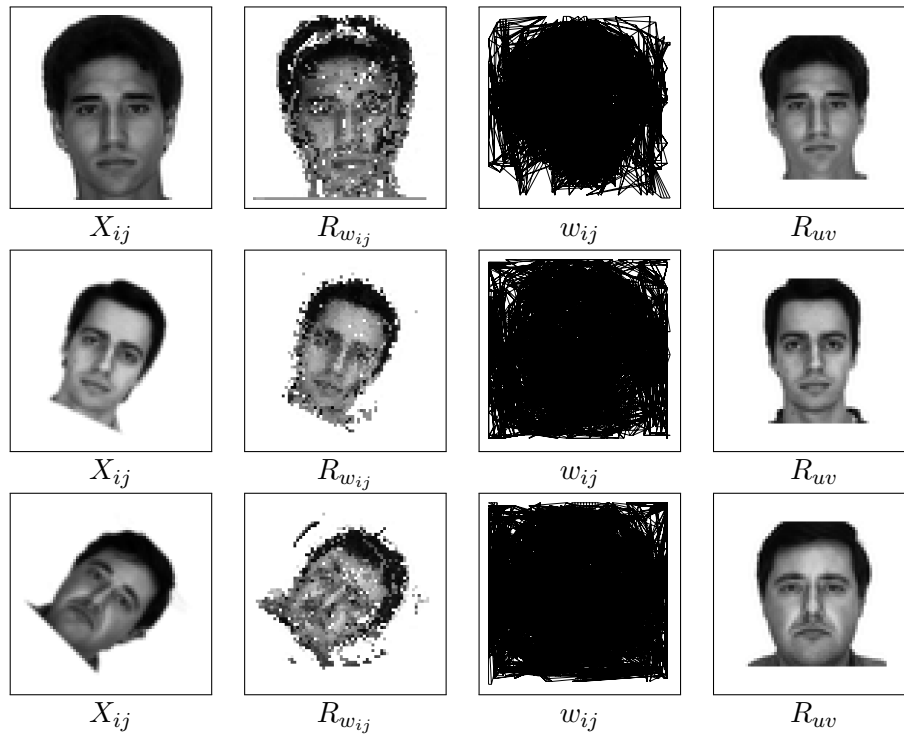


Figure 7.2: ZOW examples for images with no, 25 degrees and 45 degrees rotation.

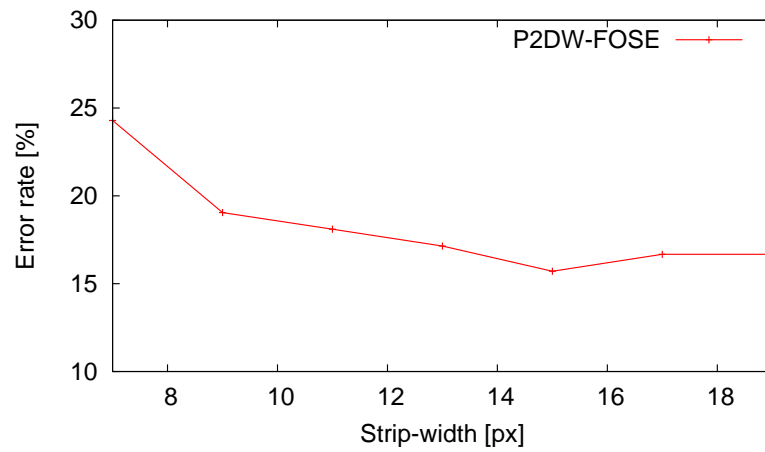


Figure 7.3: Rotated-45 AR-Face: Results of the P2DW-FOSE with different strip-width  $\Delta$ .

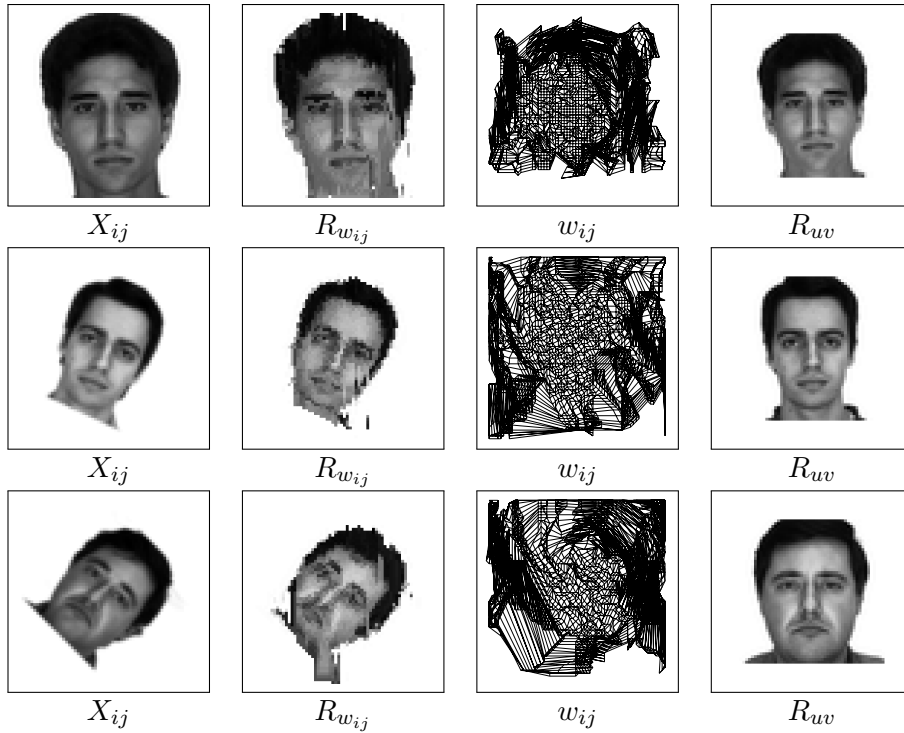


Figure 7.4: P2DW-FOSE examples for images with no, 25 degrees and 45 degrees rotation.

significantly at the image borders. Even the crossing of paths can be observed, all leading to horizontal Sakoe constraint violations. The vertical Sakoe constraints however can not be violated due to the model definition (cf. Section 3.4). Similar effects are visible in the distortion grid for the images with 25 degrees rotation. However in this case the center of distortion grid also reflects the rotation. The originally vertical and horizontal connections are now mainly diagonal. The borders are dominated by strong distortions again. For the example with a rotation of 45 degrees these effects are aggregated. Again there are mostly diagonal connections in the center of the grid, but even stronger distortions at the borders. The resulting distorted images are close to the test image, however a few artifacts are still present. The rotations and scalings in all three images are compensated.

It should be noted that in the P2DW-FOSE large horizontal deviations between pixels of two paths are in most cases not covered by the penalty function. The P2DW only assigns one penalty for the complete column and for P2DW-FOSE this penalty is defined by the center of the strips. If two pixels of two neighboring columns that have the same  $j$ -position are located at the opposing borders of the strip, the penalty is the same as if they were neighbors again. For this reason, large deviations as they are observable in the P2DW-FOSE distortion grids can not be averted by the penalty function.

### 7.3 Tree-Serial Dynamic Programming

Since the same penalty function is used in all experiments for comparability, there are no parameters for the TSDP to be optimized. The result on the Rotated-45 AR-Face database is with 13.30 % quite good compared to the other methods. On the CMU-PIE database error rates of 0.25% for near-frontal images and 7.35% for near-profile images are obtained leading to an average error rate of 3.80% [Gass & Pishchulin<sup>+</sup> 11].

Figure 7.5 shows the warpings for the three example images. Although TSDP aligns each column independently just as P2DW-FOSE, the independence is not obvious in the distortion grids. The Sakoe constraint violations are minimal, especially if no rotation is involved. In the latter case, the original structure of the grid is well maintained. Again on the borders the percentage of distortions increases, most of them without Sakoe constraint violations. The reason for this is that the alignments of the columns is strongly influenced by the lookahead defined by the forward and backward run of dynamic programming. The lookahead however was aligned using horizontal Sakoe constraints. For the two examples with rotations, the diagonal connections are visible again in the center while the borders are more and more subject to strong distortions. Also the Sakoe constraint violations increase. Still, TSDP is also able to compensate the rotations and scalings in all three cases.

Similar to P2DW-FOSE, also TSDP does not penalize horizontal deviations explicitly, as horizontal penalties are only used when the forward and backward scores are included. This has only an implicit effect on the next column, since it uses the next forward and backward scores. As result, also TSDP can not avert large horizontal deviations.

### 7.4 Sequential Tree-Reweighted Message Passing

The CTRW-S is the first method that implements the Sakoe constraints completely in both dimensions [Gass & Dreuw<sup>+</sup> 10]. The performance depends strongly on the number of iterations used as illustrated in Figure 7.6. Using only one iteration, i.e. one forward-pass and one backward-pass through the nodes, the result is an error rate of 27.62 %. Increasing the number of iterations improves the error rate such that with five iterations the error rate is already at 19.52 %. Increasing the number of iterations further has only a slight influence on the error rate, the best result being 18.10 % with 21 iterations at the cost of a very high complexity (cf. Figure 7.6). With more than 25 iterations the error rate deteriorates again. The reason for this is that with such a high number of iterations, most warpings are already very close to the lower bound. Only for images that are harder to warp, usually images of the wrong class, there is still room for improvement. This can cause some images that were classified correctly with less iterations to be classified falsely if too many iterations are used.

On the CMU-PIE pose database, the error rate is with 0.49% on near-frontal images slightly worse than for TSDP and P2DW-FOSE. However, 6.37% on near-profile images and therefore an average error rate of 3.43% denotes the best results for the state-of-the-art algorithms as

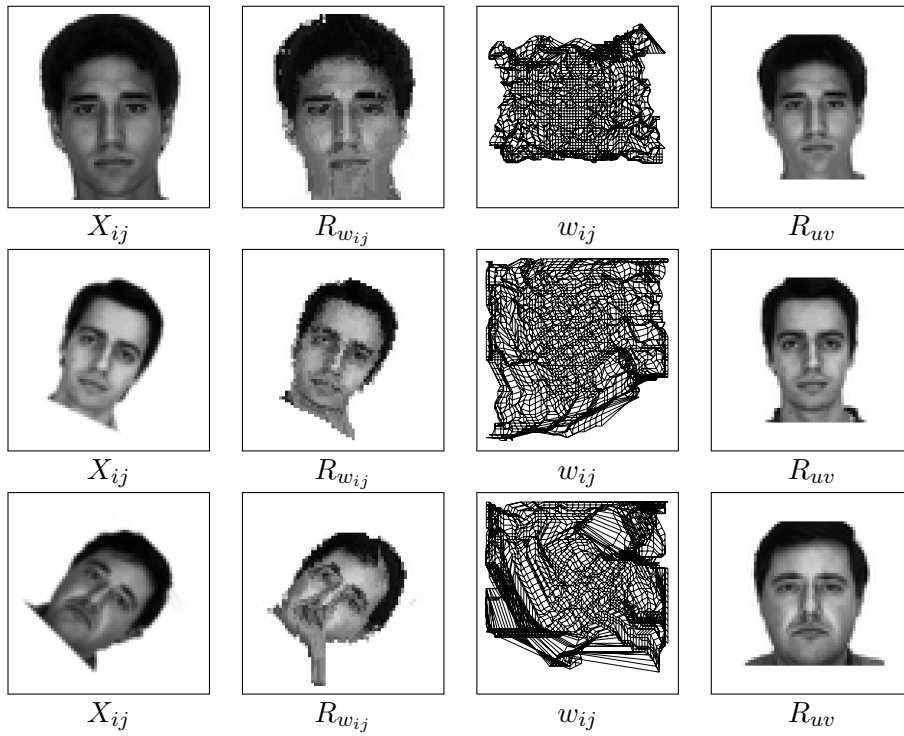


Figure 7.5: TSDP examples for images with no, 25 degrees and 45 degrees rotation.

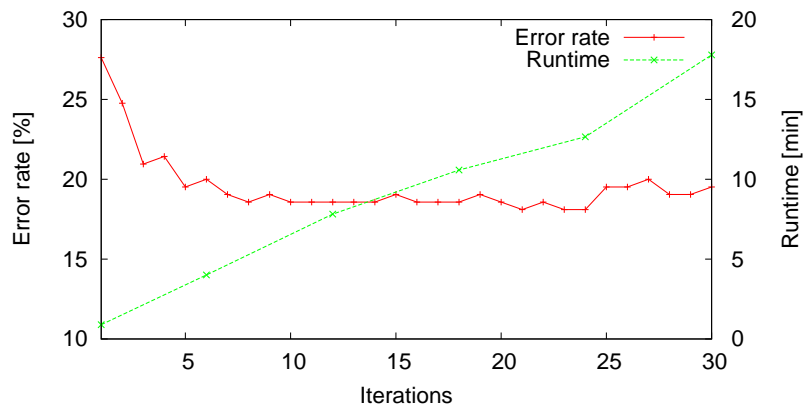


Figure 7.6: Rotated-45 AR-Face: Results of the CTRW-S with different numbers of iterations.

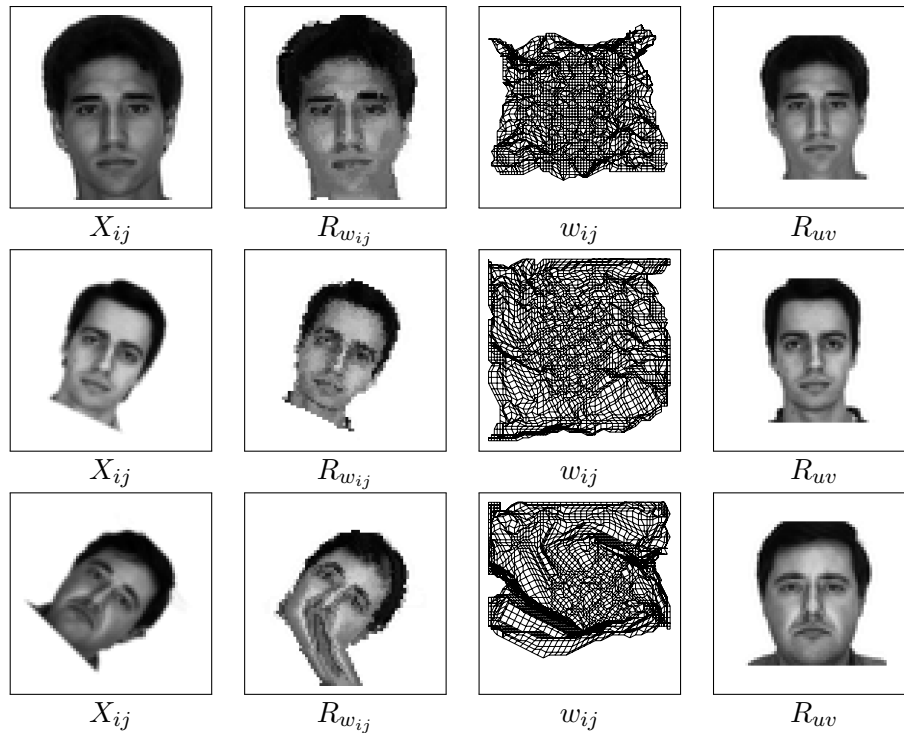


Figure 7.7: CTRW-S examples for images with no, 25 degrees and 45 degrees rotation.

shown in [Gass & Pishchulin<sup>+</sup> 11].

Analyzing the warping grids produced by the CTRW-S (cf. Figure 7.7), the tasks with scalings and medium rotation can be handled well within the Sakoe constraints. Both grids are similar to the grids generated by the TSDP (cf. Figure 7.5). Again the center of the grid corresponding to no rotation retains the original structure of the grid and the center of the grid for the image with 25 degrees rotation is dominated by the diagonal connections. In both cases, the distorted reference image comes very close to the test image. However, the distortion for the image with 45 degrees rotation leads to artifacts in the lower part of the image.

## 7.5 Constrained Tree-Serial Dynamic Programming

Since Constrained Tree-Serial Dynamic Programming (C-TSDP) extends TSDP by horizontal dependencies, the achieved scores are usually lower than the scores of the TSDP. Such a relation often also leads to a worse recognition rate. This is also the case for the C-TSDP (cf. Table 7.2). TSDP optimizes each column independently, such that the local decisions for one column do not affect the decisions for the other columns. For the C-TSDP the alignment of each column except the starting column  $i_s$  depends on the alignment of the previous columns.

Table 7.2: Rotated-45 AR-Face: Results of the C-TSDP compared to the TSDP.

Method	$i_s$	Error rate [%]	Abs. errors	Runtime [s]
TSDP	-	13.30	28	24.47
C-TSDP	$I/2$	30.48	64	21.97
C-TSDP	1	31.90	67	21.80
C-TSDP-ALA	$I/2$	22.86	48	38.67
C-TSDP-ALA	1	20.95	44	38.72

The local decisions that are only optimal for the column at hand, influence the alignment of the next columns. If the local decisions are globally far from optimal, the compensation can only happen slowly, since due to the Sakoe constraints the allowed pixel skips are limited.

Table 7.2 also shows the effect of the aligned lookahead. C-TSDP uses by definition the same lookahead as TSDP. However, applying the aligned lookahead instead, the recognition result is improved significantly from 30.48% to 22.86%. The effect of both lookaheads will be further examined in context of Two-Level Dynamic Programming (cf. Section 7.6). The table also contains the results for the C-TSDP when the first column is chosen as starting column  $i_s$ . As mentioned in Section 4.8, this can be seen as a baseline for the 2LDP-LA that will be investigated in the next section. While for the basic C-TSDP choosing the middle column as starting point yields slightly better results, when the aligned lookahead is applied, the opposite effect can be observed. However, in both cases, the error rate of TSDP is not reached, due to the better scores achieved by TSDP. As mentioned in Section 4.8.2, C-TSDP has a reduced complexity compared to TSDP. This is also reflected in the runtimes. However, since the factors in Table 4.2 are only approximations, they are not reproduced exactly. Additionally, there is an offset caused by the distance caching Section 5.1. It is also observable that the runtime does not depend on the starting column  $i_s$ .

Figure 7.8 illustrates the effect of the aligned lookahead on the scores. The figure contains score difference histograms, i.e. for each comparison of two images during one experiment, the achieved scores are subtracted and a histogram is generated over the subtraction results. In Figure 7.8 the scores of the C-TSDP-ALA are subtracted from the scores of the C-TSDP. If the result is positive, then the score of the C-TSDP-ALA was lower and therefore the model found a better solution to the minimization problem. This is done once for warpings that are computed from two images with the same class leading to the within-class histogram (cf. Figure 7.8(a)) and once for warpings that are computed from two images with different classes leading to the between-class histogram (cf. Figure 7.8(b)). In general, scores achieved on the Rotated-45 AR-Face database range from 2000 to 4000. In both cases it is observable that the aligned lookahead improves the performance of the C-TSDP, the scores are mostly decreased.

In Figure 7.9 the scores of C-TSDP-ALA are compared to the scores of CTRW-S, the latter are subtracted from the former. For CTRW-S the results after 21 iterations are considered. Also in this case, CTRW-S improves the scores of C-TSDP-ALA, however not that signifi-

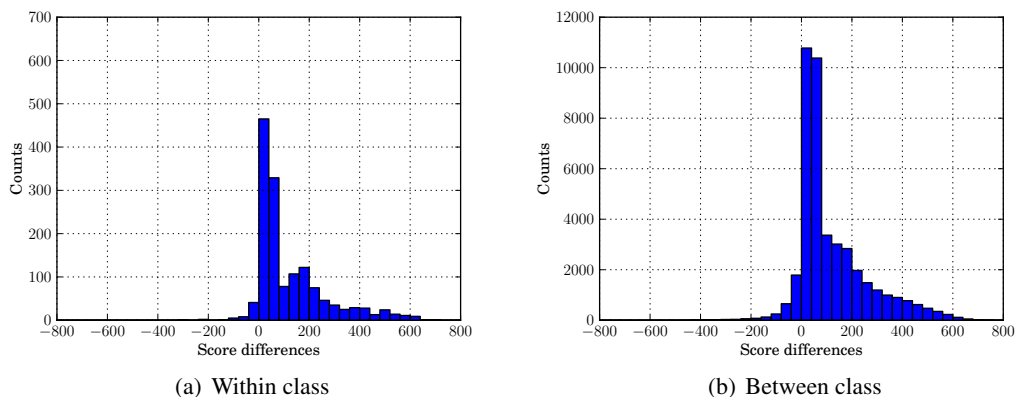


Figure 7.8: Rotated-45 AR-Face: Score difference histograms to compare the scores of C-TSDP with the score of C-TSDP-ALA. The scores of C-TSDP-ALA are subtracted from the scores of C-TSDP.

cantly. In some cases, C-TSDP-ALA even outperforms CTRW-S.

In Table 7.3 the results for the CMU-PIE pose database are reported. Since for the CMU-PIE pose database the mapping is reversed and the reference images are mapped onto the test images, the starting column  $i_s$  is now defined on the reference image. The most remarkable observation is that the aligned lookahead leads to a worse error rate than the basic lookahead. The reason for this can be explained by the help of the score difference histograms in Figure 7.10 for the near-profile subset. The histograms are generated by subtracting the scores of the C-TSDP-ALA from the scores of the C-TSDP. On the CMU-PIE database, the scores usually range from 1000 to 3000. Both the between-class (cf. Figure 7.10(b)) and within-class (cf. Figure 7.10(a)) histograms show that the vast majority of the score differences is positive. Thus the scores are still improved by using the aligned lookahead. However, the negative effect on the recognition rate is rooted in the fact that mostly the between-class scores profit from the improvement. While for the within-class scores, the differences are mostly close to zero, the between-class histograms tends more into the positive area. As a result, for some images the scores for matching the test image to a wrong classes are improved too much, such that the scores resulting from matching the wrong class exceed the score for the right class.

Figure 7.11 shows how the C-TSDP handles the different scaling and rotation challenges. The starting column in all cases is  $i_s = I/2$  and is marked red in the distortion grids. The warping grids for no rotation and medium rotation expose the same qualities that can be observed for the other methods. The grid corresponding to no rotation is oriented at the original 2D grid, the center of the grid corresponding to 25 degrees rotation is dominated by diagonal connections. In both cases, the challenges are fully compensated. The strong rotation of 45 degrees however can only be handled partially. The alignment of the starting column is mapped

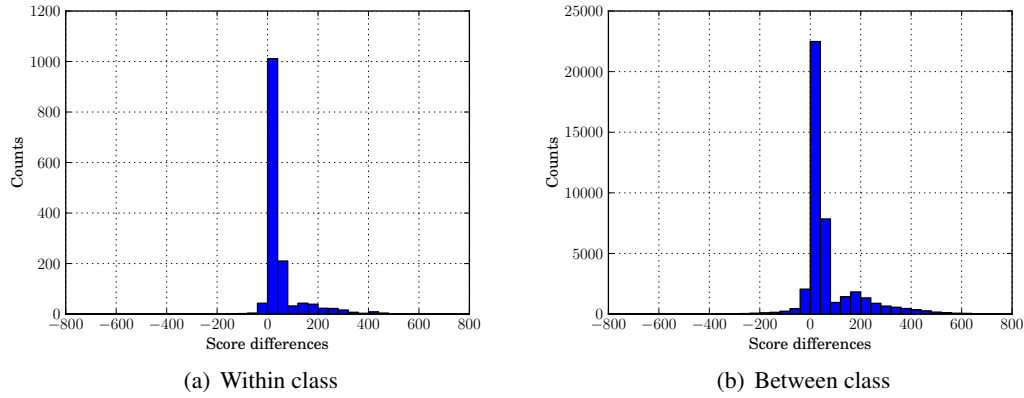


Figure 7.9: Rotated-45 AR-Face: Comparison of C-TSDP-ALA and CTRW-S. The scores of CTRW-S are subtracted from the scores of C-TSDP-ALA.

Table 7.3: CMU-PIE: Results of the C-TSDP compared to the TSDP.

Method	$i_s$	Frontal ER [%]	Profile ER [%]	Total ER [%]	Abs. errors
TSDP	-	0.25	7.35	3.80	31
C-TSDP	1	0.25	7.11	3.68	30
C-TSDP	$I/2$	0.25	9.31	4.78	39
C-TSDP-ALA	1	0.25	9.80	5.02	41

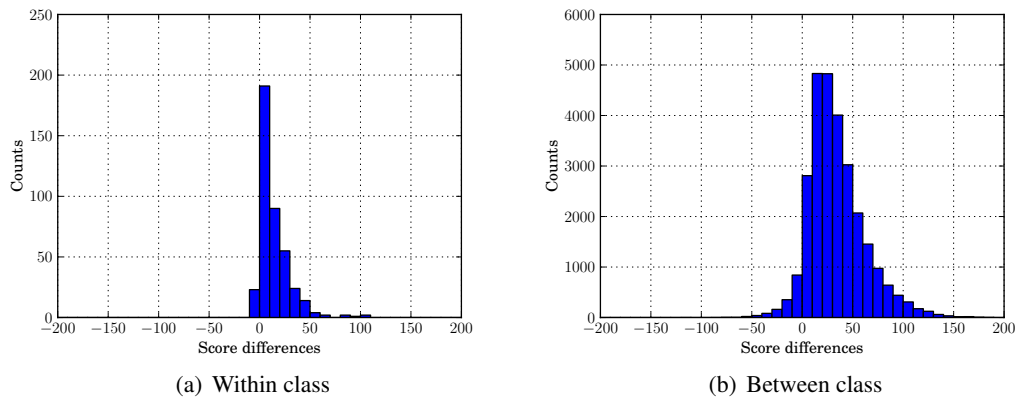


Figure 7.10: Score difference histograms comparing the C-TSDP and the C-TSDP-ALA on the near-profile images of the CMU-PIE database. The scores of the C-TSDP-ALA are subtracted from the scores of the C-TSDP.

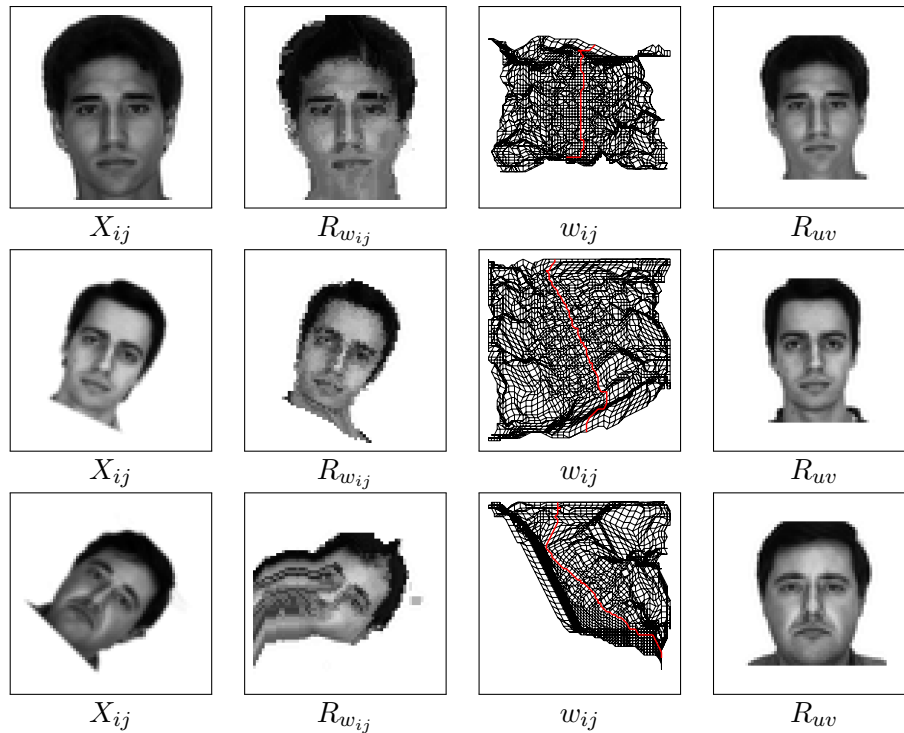


Figure 7.11: C-TSDP examples for images with no, 25 degrees and 45 degrees rotation. The mapping of the starting column  $i_s$  is marked red.

optimally according to the lookahead, but the fixed decision for this mapping does not leave enough options for the other columns. The right part of the image can still be handled, for the left part the algorithm is not able to find good alignments within the Sakoe constraints leading to the artifacts. While the TSDP (cf. Figure 7.5) is able to compensate the decision for the middle column by using large deviations between neighboring columns, the C-TSDP is bound to the Sakoe constraints and has to align each next column close to the previous one. In some cases such as the example for 45 degrees rotation, the maximal pixel skip of one defined by the Sakoe constraints is not enough to compensate globally less than optimal decisions for the starting column.

## 7.6 Two-Level Dynamic Programming

In this section the results of Two-Level Dynamic Programming and its various extensions and modifications will be analyzed.

Table 7.4: Rotated-45 AR-Face: Results of 2LDP with and without lookaheads

Method	Error rate [%]	Abs. errors	Runtime [s]
2LDP	47.14	99	48.40
2LDP-LA	27.62	58	52.47
2LDP-ALA	20.95	44	72.57

### 7.6.1 Lookahead

The evaluation of 2LDP is started by investigating the effect of the lookahead on the performance of the algorithm. For the following experiments, a lookahead weight of  $\gamma = 1.0$  is selected. Results for different values are given in Section 7.6.5. As representative  $j^* = J/2$  is selected as indicated in Section 4.2. For this parameter an analysis follows in Section 7.6.6.

The results for the Rotated-45 AR-Face database are reported in Table 7.4. The basic version of 2LDP reaches an error rate of 47.14%, which is rather high. However, by adding the lookahead the error rate is improved significantly. For 2LDP the columns are optimized starting with the first column of the image. During this optimization, the best decisions only for this column are made. This can be greedy, even though several candidates are considered. Globally not optimal candidates are hard to compensate within the small deviations allowed by the Sakoe constraints. With the lookahead, also the rest of the image is included in each decision, also for the first column. The alignments are therefore closer to the globally best solution, ultimately leading to a better recognition performance.

The improvement of the scores is illustrated in Figure 7.12, where the score difference histograms comparing 2LDP and 2LDP-LA are shown. The range of the scores achieved by the two variants is from 2000 to 4000. For both, the between and the within class histogram the score differences are far in the positive area, meaning the scores are strongly reduced by adding the lookahead. In Figure 7.13 the scores generated by 2LDP-LA and 2LDP-ALA are compared. Also in this case the differences are mostly positive, meaning 2LDP-ALA improves the scores further compared to 2LDP-LA. However, the impact is not as strong as for 2LDP and 2LDP-LA.

On the CMU-PIE database (cf. Table 7.5) the same effect can be observed. Including the lookahead encourages the alignment of the first columns to be oriented on the global optimum. This results again in better overall warpings and an improved error rate. However, similar to C-TSDP (cf. Section 7.5), using the aligned lookahead (ALA) instead of the basic lookahead (LA) has no positive effect on the error rate, due to an improved warping of between-class images.

In Figure 7.14 the warping grids and distorted images generated by 2LDP for the three examples with different rotations are shown. The mappings of the representatives are marked red. The first example with no rotation can be handled well by 2LDP. The 2D grid structure is still recognizable in the deformation grid and the representatives are mapped almost to the same row in the reference image. Also the resulting distorted image is close to the test image.

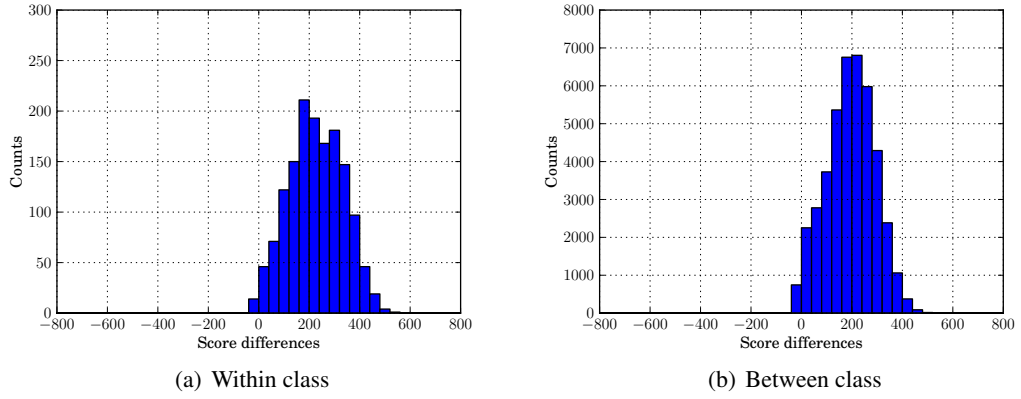


Figure 7.12: Rotated-45 AR-Face: Comparison of 2LDP and 2LDP-LA. The scores of 2LDP-LA are subtracted from the scores of 2LDP.

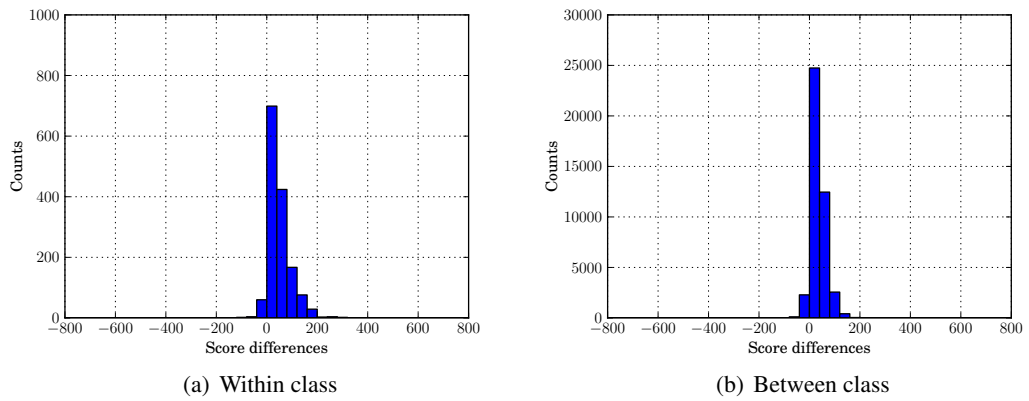


Figure 7.13: Rotated-45 AR-Face: Comparison of 2LDP-LA and 2LDP-ALA. The scores of 2LDP-ALA are subtracted from the scores of 2LDP-LA.

Table 7.5: CMU-PIE: Results of 2LDP with and without lookahead.

Method	Frontal ER [%]	Profile ER [%]	Total ER [%]	Abs. errors
2LDP	2.45	16.67	9.56	78
2LDP-LA	0.25	9.07	4.66	38
2LDP-ALA	0.25	9.80	5.02	41

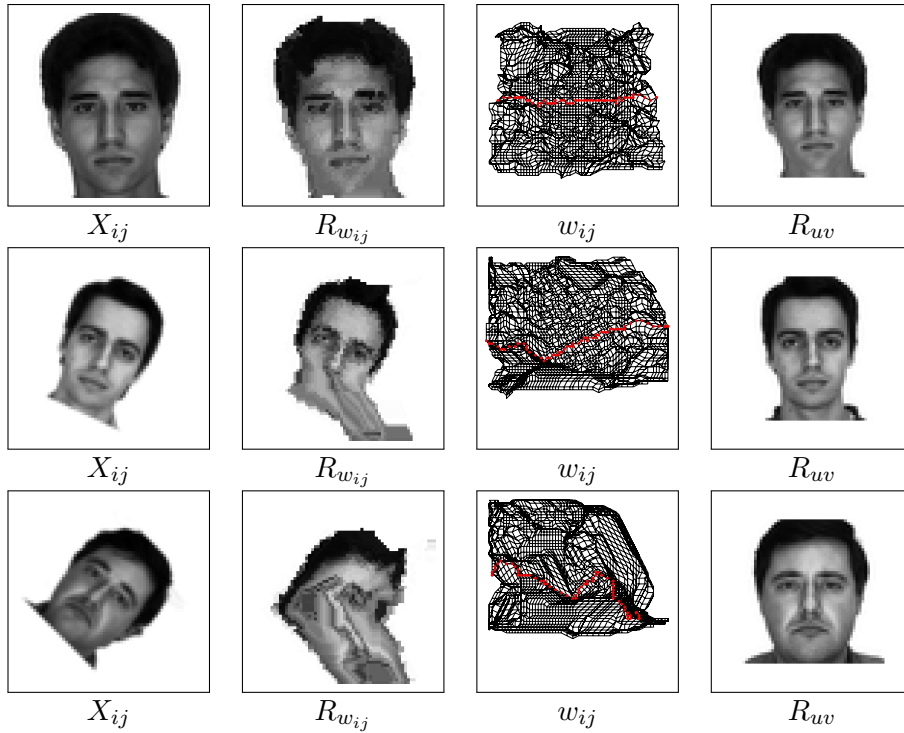


Figure 7.14: 2LDP examples for images with no, 25 degrees and 45 degrees rotation. The representative is set to  $j^* = J/2$  and their mappings are marked red.

For the two examples with rotations, the drawbacks of 2LDP stand out. The alignments are computed from the left to the right and work well for the first third of the image. However, when arriving at the middle and the last third of the image, it becomes evident that the candidates computed for the first columns are too greedy and can not be compensated in the rest of the image leading to strong distortions. The 2LDP is therefore not able to reconstruct the test image.

The flaws of 2LDP evident in Figure 7.14 are addressed by adding the lookahead leading to significant improvements as can be seen in Figure 7.15. The latter contains the deformation grids and distorted images for 2LDP-ALA. The image with no rotation was already well aligned using no lookahead. With the lookahead a similar deformation is computed. For the image with 25 degrees rotation, the 2LDP-ALA is able to compute a good alignment as well. The representatives are mapped to form a slightly diagonal line from the bottom left to the top right, which is the expected result due to the rotation. This was already visible for the 2LDP, however only partially. Although also the distortion for the image with 45 degrees rotation is improved, there are still artifacts present. In this case even with the lookahead, the candidates computed for the first columns were not globally optimal for the bottom half of the image.

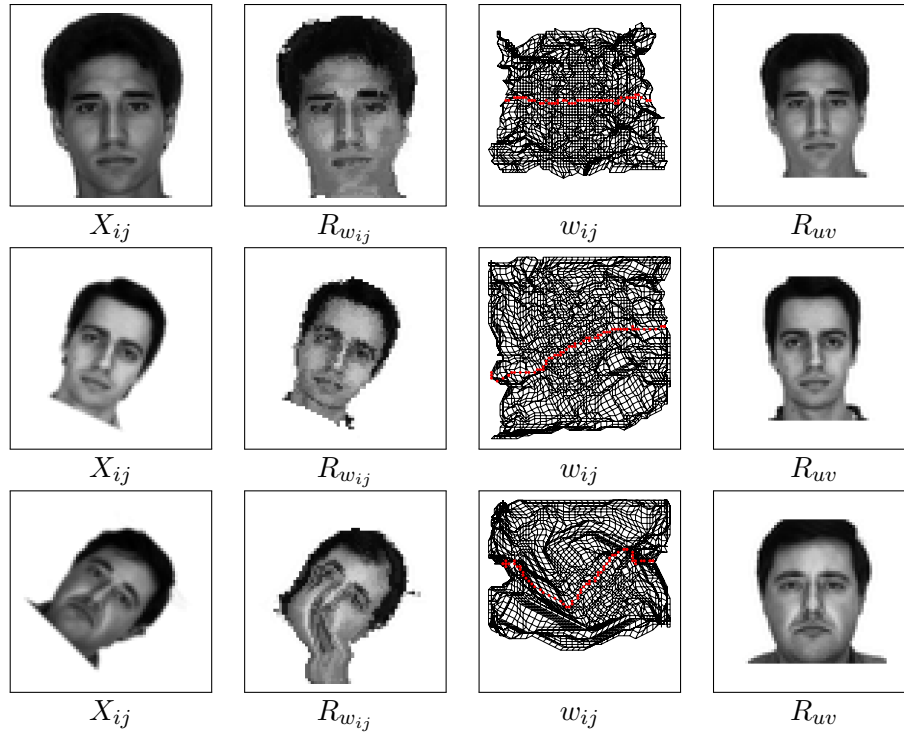


Figure 7.15: 2LDP-ALA examples for images with no, 25 degrees and 45 degrees rotation. The representative is set to  $j^* = J/2$  and their mappings are marked red.

However, the top half is still aligned well and for the second half of the image, the expected diagonal structure for the representatives is observable.

In summary, for all cases the lookahead leads to better scores and the aligned lookahead improves the scores further, which can lead to an unwanted improved warping of between-class images. As a result, 2LDP-LA improved the recognition performance on both databases, while with 2LDP-ALA only the error rate on the difficult Rotated-45 AR-Face database is improved.

### Comparison with C-TSDP

As mentioned in Section 4.8, the C-TSDP with starting column  $i_s = 1$  can be interpreted as a baseline of 2LDP-LA without the global optimization. Comparing the results for the two methods on the Rotated-45 AR-Face database (cf. Table 7.2 and Table 7.4), 2LDP-LA reduces the error rate by 4%. This means, adding the optimization over several candidates for each column alignment pays off and better results are obtained. However, considering the 2LDP-ALA and the C-TSDP-ALA as baseline, the recognition results are the same. Here, adding the global optimization does not improve the recognition performance. The reason

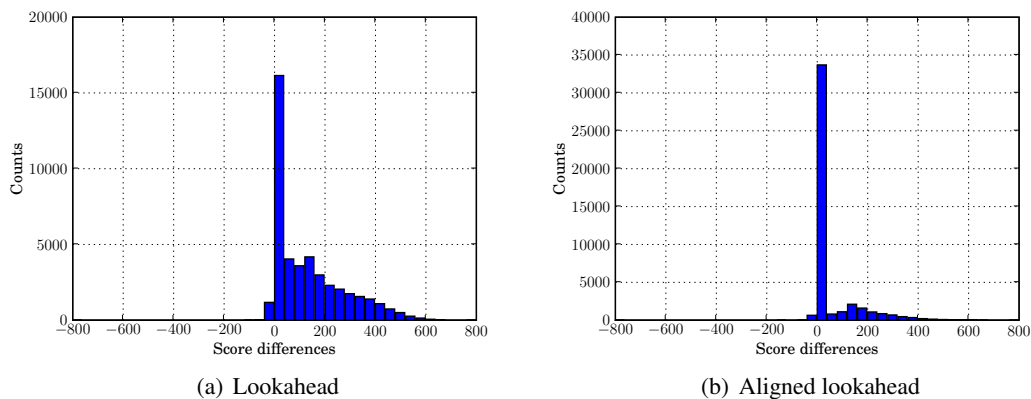


Figure 7.16: Rotated-45 AR-Face: Comparison of C-TSDP and 2LDP with lookahead. In (a) the 2LDP-LA scores are subtracted from the C-TSDP scores, in (b) the 2LDP-ALA scores are subtracted from the C-TSDP-ALA scores.

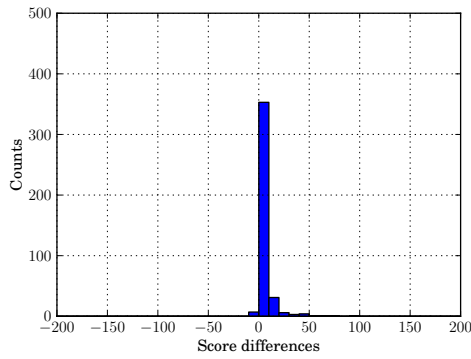
for this is observable in Figure 7.16. While for the basic lookahead the scores are often improved (cf. Figure 7.16(a)) by using 2LDP instead of C-TSDP, for the aligned lookahead (cf. Figure 7.16(b)) the improvement is minimal.

On the CMU-PIE pose database, the error rate of the 2LDP-LA is even worse than the error rate of the C-TSDP, even though the scores themselves are not. As shown in Figure 7.17, the same effect as for the C-TSDP and the C-TSDP-ALA (cf. Section 7.5) on this database occurs. Although, the scores are better, the recognition is worse, since mostly the between class warpings profit from the improvement.

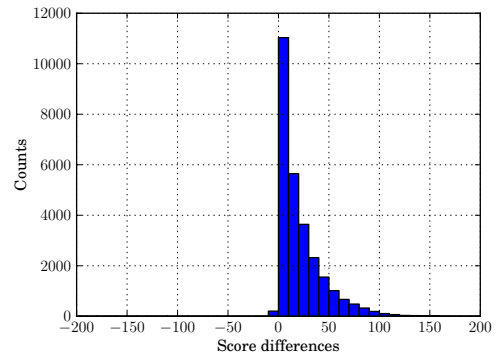
In general, the scores achieved by the 2LDP-LA and 2LDP-ALA are better than the scores achieved by the C-TSDP and C-TSDP-ALA (cf. Figure 7.16 and Figure 7.17). However, this is not always reflected in the recognition performance. For the difficult Rotated-45 AR-Face database, the 2LDP-LA also achieves a better error rate than the C-TSDP, however with the aligned lookaheads, the error rates are equal. On the CMU-PIE on the other hand, the C-TSDP has the best recognition performance, since for 2LDP-LA the results for between-class warpings are too good.

## Comparison with CTRW-S

Comparing the distortion grids of 2LDP-ALA (cf. Figure 7.15) and CTRW-S (cf. Figure 7.7) it is observable that the grids and therefore the distorted images look very much alike. Also on score level, both methods achieve similar results, but CTRW-S generates warpings with slightly less costs (cf. Figure 7.18).

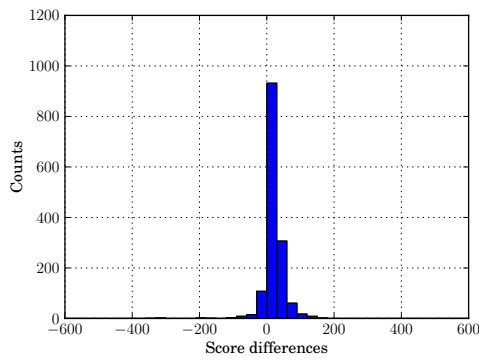


(a) Within class

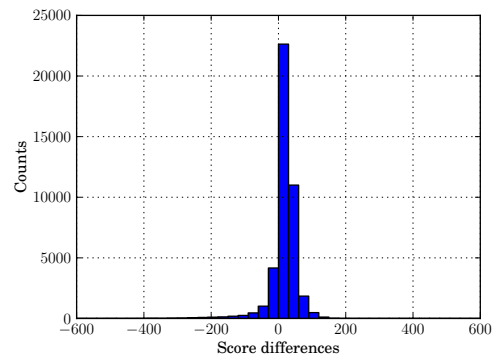


(b) Between class

Figure 7.17: CMU-PIE: Comparison of C-TSDP and 2LDP-LA. The scores of 2LDP-LA are subtracted from the scores of C-TSDP.



(a) Within class



(b) Between class

Figure 7.18: Rotated-45 AR-Face: Comparison of 2LDP-ALA and CTRW-S. The scores of CTRW-S are subtracted from the scores of 2LDP-ALA.

Table 7.6: Rotated-45 AR-Face: Results of 2LDP compared to 2LDP-U with different lookaheads.

Method	Error rate [%]	Abs. errors	Runtime [s]
2LDP	47.14	99	48.40
2LDP-U	54.29	114	15.81
2LDP-LA	27.62	58	52.47
2LDP-U-LA	26.67	56	21.82
2LDP-ALA	20.95	44	72.57
2LDP-U-ALA	20.95	44	40.03

## 7.6.2 Restriction to $u$ -component

In this section the effect of optimizing only over the  $u$ -component of the representative is investigated. As can be seen in Table 7.6 also in this case adding the lookahead improves the recognition performance and by using the aligned lookahead the error rate deteriorates even further. However, comparing the results of Table 7.6 with the results of 2LDP there is an interesting observation. While for the basic variant without lookahead 2LDP achieves the expected better result than 2LDP-U, when the lookahead is included, 2LDP and 2LDP-U obtain almost the same error rates. The reason for this is the similarity of the candidates that are computed for each column. Already without the lookahead it is likely that the best path through pixel  $(u, v)$ , i.e. the best path where the representative is mapped to  $(u, v)$ , will be similar to the best path going through the direct neighbors of  $(u, v)$ . This leads to very similar candidates. The difference of 2LDP-U compared to 2LDP is that less candidates are computed for each column. While for the basic 2LDP the variability in the candidate paths is sufficient to make the reduction by 2LDP-U noticeable, this is not the case anymore for the variants with the lookahead. The lookahead restrains the variability of the candidates paths further, such that leaving out some of them does not affect the optimization significantly.

These observations are not only reflected by the recognition performance, but also by the scores themselves. For the score difference histograms in Figure 7.19 the scores of 2LDP are subtracted from the scores of 2LDP-U. The histograms are pooled, there is no differentiation between within-class or between-class scores. When no lookahead is used (cf. Figure 7.19(a)), 2LDP clearly outperforms 2LDP-U. However, when the lookahead is added (cf. Figure 7.19(b)) the scores achieved by both methods are almost equal.

In conclusion, with the lookahead many of the candidates 2LDP computes are very similar. Therefore, not all of them are needed and also with less candidates as used by 2LDP-U similar results are obtained.

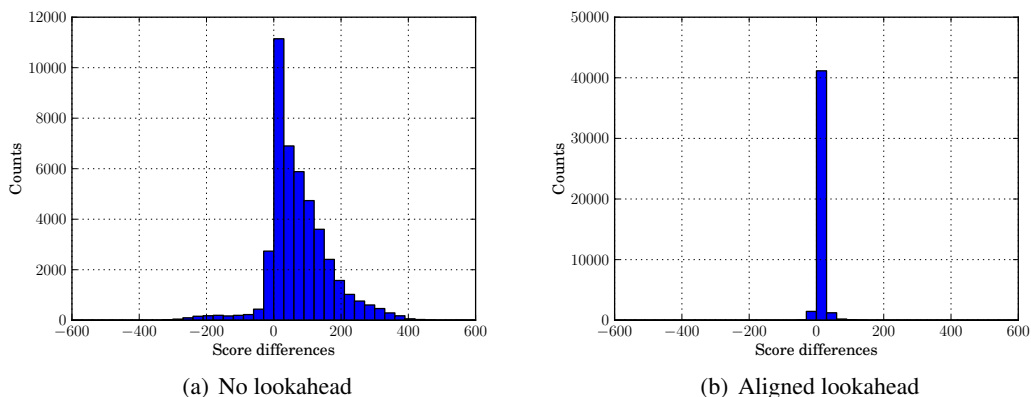


Figure 7.19: Rotated-45 AR-Face: Comparison of 2LDP and 2LDP-U with and without lookahead. The scores of 2LDP are subtracted from the scores of 2LDP-U.

Table 7.7: Rotated-45 AR-Face: Results of 2LDP with the local-best strategy.

Method	Error rate [%]	Abs. errors	Runtime [s]
2LDP	47.14	99	48.40
2LDP-LB	49.05	103	44.50
2LDP-ALA	20.95	44	72.57
2LDP-LB-ALA	20.95	44	64.89

### 7.6.3 Local-best selection strategy

A similar conclusion as for 2LDP-U can be drawn when analyzing the results for the local-best selection strategy. Table 7.7 contains the results of using the local-best selection strategy during the optimization of 2LDP (2LDP-LB). While computing the scores without lookahead leads to a slightly better result for 2LDP (cf. Table 7.4), with the lookahead the same result is achieved by both methods. This indicates that the paths selected using 2LDP or 2LDP-ALA are mostly the same paths that 2LDP-LB is forced to use.

Also the score difference histograms in Figure 7.20, where the scores of 2LDP-LB are subtracted from the scores of 2LDP, show that 2LDP-LB achieves similar results as 2LDP. Already with no lookahead, there is only a moderate improvement of the scores by using the full optimization. With the lookahead, the differences in the scores cluster around zero.

### 7.6.4 Strip-Restriction

The idea of the strip-restriction is to restrain the possible warpings when a good alignment of the objects that are compared can be assumed. This is obviously not the case for the rotated

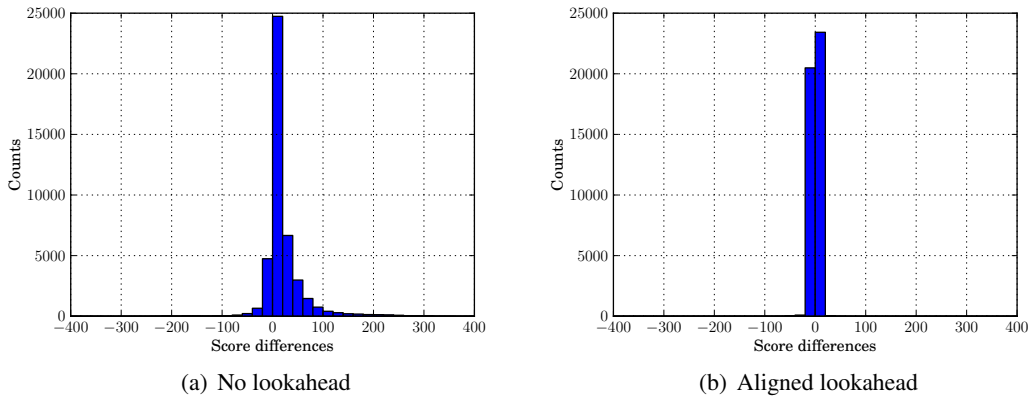


Figure 7.20: Rotated-45 AR-Face: Comparison of 2LDP and 2LDP-LB with and without lookahead. The scores of 2LDP are subtracted from the scores of 2LDP-LB.

Table 7.8: Rotated-45 AR-Face: Results of the 2LDP with and without strip-restriction.

Method	Error rate [%]	Abs. errors	Runtime [s]
2LDP	47.14	99	48.40
2LDP-S	44.76	94	94.69
2LDP-ALA	20.95	44	72.57
2LDP-S-ALA	22.38	47	115.94

faces in the Rotated-45 AR-Face database. This has to be compensated by a very wide strip-width. For P2DW-FOSE the value  $\Delta = 15$  is optimal (cf. Section 7.2). Thus also for 2LDP-S experiments  $\Delta = 15$  is selected. As can be observed in Table 7.8, without lookahead, adding the strip-restriction improves the error rate. In this case, the restraint on relatively vertical paths gives the algorithm a weak orientation for the optimization, which is not the case for the basic 2LDP, where the paths can be aligned arbitrarily. However, with the lookahead 2LDP has a very good orientation for the optimization, rendering the strip-restriction obsolete. Moreover, in this case the restraint even leads to a worse result, since it is not a good assumption for rotated objects.

In contrast to the Rotated-45 AR-Face database the images in the CMU-PIE database are well aligned. Therefore 2LDP-S is also tested on this database. Again, the same strip-width as for P2DW-FOSE is used, i.e.  $\Delta = 3$ . However, as can be seen in Table 7.9, the error rate is not improved.

Table 7.9: CMU-PIE: Results of the 2LDP compared to the 2LDP-S.

Method	Frontal ER [%]	Profile ER [%]	Total ER [%]	Abs. errors
2LDP-LA	0.25	9.07	4.66	38
2LDP-S-LA	0.25	10.05	5.15	42

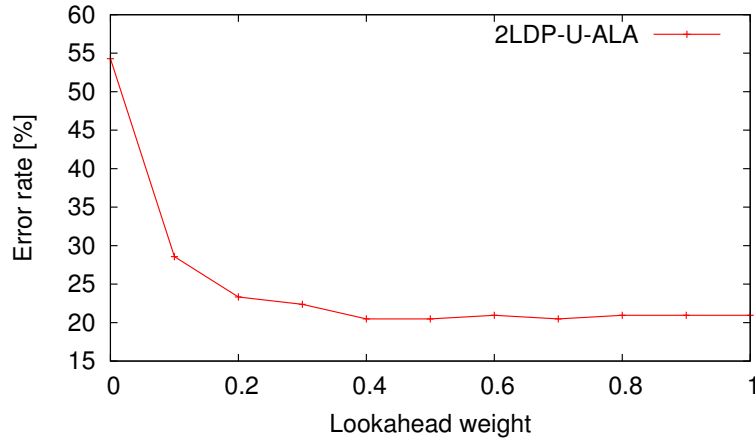


Figure 7.21: Rotated-45 AR-Face: Results for 2LDP-U-ALA with different values for  $\gamma$ .

### 7.6.5 Lookahead weight

As mentioned in Section 4.6 and Section 7.6.1, the contribution of the lookahead can be weighted by a factor  $\gamma$ . Since 2LDP-U-ALA achieved the same results as 2LDP-ALA, the effect of the weight factor is analyzed using 2LDP-U-ALA for efficiency. The results are given in Figure 7.21. Starting with a weight of 0.0, which corresponds to using no lookahead, the error rate improves rapidly until a value of 0.4 is reached. For higher weights the error rate remains almost the same. Since for all values greater than zero, the complexity of the algorithm does not depend on  $\gamma$  it is therefore reasonable to use the weight of  $\gamma = 1.0$ .

### 7.6.6 Representative

In the above experiments usually the middle row of the test image was selected for the representatives, i.e.  $j^* = J/2$ . The plot in Figure 7.22 shows results for other values as representative. Again, 2LDP-U-ALA is selected for efficiency. It is observable that the performance of the algorithm varies only slightly with different representatives, which is the desired result. While  $j^* = J/2$  yields an error rate of 20.95%, with  $j^* = 1$  and  $j^* = J$  the error rate improved. However in absolute numbers, this difference corresponds to only one or two images.

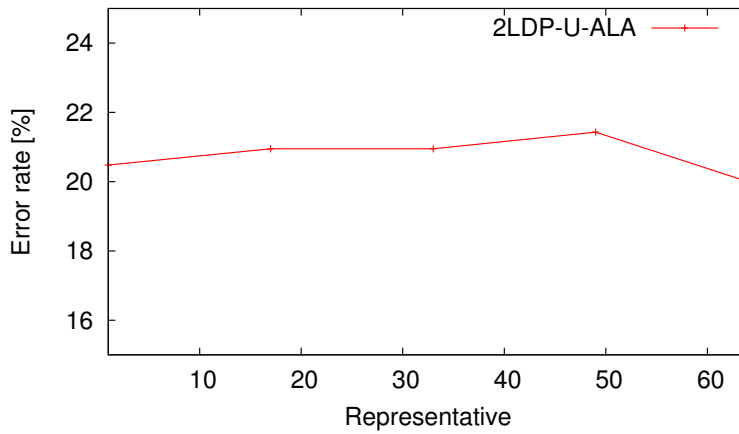


Figure 7.22: Rotated-45 AR-Face: Results of 2LDP-ALA with different representatives.

## 7.7 Nearest Neighbor Pruning

As mentioned in Section 5.3, the classification by the 1-Nearest Neighbor (1-NN) rule can be exploited to do pruning. In this section the effect of the pruning is evaluated using the experiment of applying 2LDP-ALA on the Rotated-45 AR-Face database (cf. Table 7.4). During the classification, each test image is compared to each reference image. This is done sequentially for each test image, i.e. the calculation of the warping for test image  $X$  and the second reference image  $R_2$  is started, when the calculation for test image  $X$  and the first reference image  $R_1$  is completed and so on. In order to profit most from applying the nearest neighbor pruning, the reference images are sorted according to the Euclidean distance to test image  $X$ . This is a very cheap operation and it is likely that images with lower Euclidean distances also achieve lower distances with the warping algorithms. In Figure 7.23 the effect of the nearest neighbor pruning is illustrated for the experiment mentioned above. The histogram is calculated by analyzing after which column of the  $64 \times 64$  the optimization was cancelled when the pruning is enabled. The bar in the right for the value 64 corresponds to the comparisons, where the optimization is not cancelled, while the bar in the left for the value zero represents the old pruning, where the computation is cancelled during the distance caching. The histogram clearly shows that recomputing the lower bound after each column optimization is profitable, since for most optimizations the algorithm can be stopped early. On average, this can be done after the 37th column, which is almost half the image.

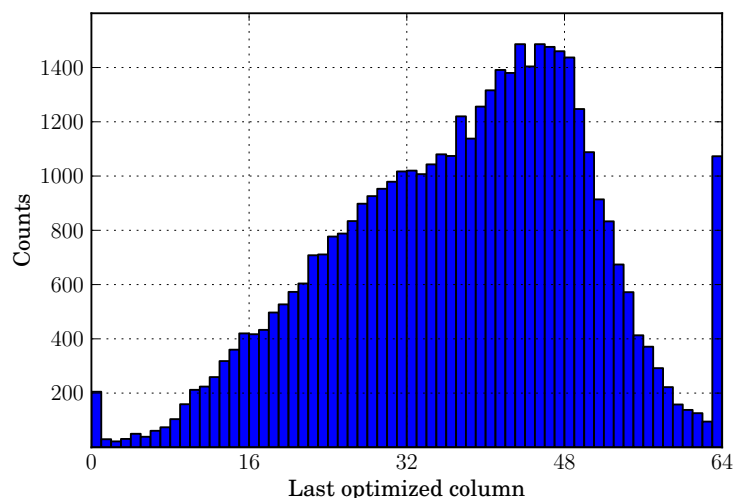


Figure 7.23: Rotated-45 AR-Face: The effect of using nearest neighbor pruning. The histogram illustrates, how often after which column the optimization is terminated. A value of zero corresponds to a termination during the distance caching. A value of 64 corresponds to a full optimization meaning no pruning could be applied.

## 7.8 Summary of the Results

In this section, the best results regarding the recognition performance are collected for the two databases.

### 7.8.1 Rotated-45 AR-Face

An overview of the results on the Rotated-45 AR-Face database is given in Table 7.10. The error rate of 90.48% when only the SIFT-Features and no warping is used illustrates the difficulty of the database. The best result was achieved by using TSDP leading to an error rate of 13.30%. The results of the algorithms implementing the Sakoe constraints are only by a few images apart. The error-rate of CTRW-S was not improved by the new methods. However, the 2LDP with 20.00% and the C-TSDP with 20.95% are both close to the 18.10% achieved by CTRW-S. Additionally, both methods are much more efficient.

### 7.8.2 CMU-PIE

The results on the CMU-PIE database are summarized in Table 7.11. On this database, the best results were achieved by the methods using the Sakoe constraints. Thereby, CTRW-S stand out with a 3.43% error rate that is not reached by any other method. The second

Table 7.10: Rotated-45 AR-Face: Overview of the results.

Method	Error rate [%]	Abs. errors	Runtime [s]
No warping	90.48	190	$3.2 \cdot 10^{-3}$
ZOW, W=32	20.00	42	7.78
P2DW-FOSE, $\Delta = 15$	15.71	33	100.23
TSDP	<b>13.30</b>	<b>28</b>	24.47
C-TSDP-ALA, $i_s = 1$	20.95	44	38.72
2LDP-ALA, $j^* = J$	20.00	42	72.57
CTRW-S, 21 iterations	18.10	38	706.73

best method is C-TSDP with 3.68% which is even slightly better than the error rate achieved with TSDP. Other than on the Rotated-45 AR-Face database, ZOW is not competitive. Also the P2DW-FOSE can not achieve the good error rates of the other methods. Next to TSDP and CTRW-S also the novel algorithms outperform other state-of-the-art approaches such as the hierarchical matching proposed in [Arashloo & Kittler 09] and the 3D shape modeling proposed in [Zhang & Gao<sup>+</sup> 08].

Table 7.11: CMU-PIE: Overview of the results.

Method	Frontal ER [%]	Profile ER [%]	Total ER [%]	Abs. errors
No warping	86.76	40.44	63.60	519
ZOW	0.49	31.61	16.05	131
P2DW-FOSE, $\Delta = 3$	<b>0.25</b>	17.63	8.94	73
TSDP	<b>0.25</b>	7.35	3.80	31
C-TSDP, $i_s = 1$	<b>0.25</b>	7.11	3.68	30
2LDP-LA, $j^* = J/2$	<b>0.25</b>	9.07	4.66	38
CTRW-S	0.49	<b>6.37</b>	<b>3.43</b>	<b>28</b>
Hierarchical Matching [Arashloo & Kittler 09]	1.22	10.30	5.76	47
3D shape modeling [Zhang & Gao <sup>+</sup> 08]	*0.00	*14.40	*6.55	*49

\*One near-profile pose is removed from the test set and added to the training set.

## Chapter 8

### Conclusion

In this thesis two novel warping algorithms with the goal to maintain full two-dimensional dependencies were introduced. The latter allows to apply geometric constraints such as the Sakoe constraints. The first novel algorithm called Constrained Tree-Serial Dynamic Programming (C-TSDP) is a modification of Tree-Serial Dynamic Programming (TSDP) that differs from TSDP by maintaining full two-dimensional dependencies allowing the model to enforce the Sakoe constraints. In this model, columns are warped sequentially using the fixed decisions for the neighboring columns.

As second algorithm Two-Level Dynamic Programming (2LDP) was introduced. Also this method maintains full two-dimensional dependencies and enforces the Sakoe constraints. In contrast to C-TSDP, 2LDP adds a global dynamic programming level to compute several candidate warpings for the columns. To overcome exponential complexity, the optimization is done only over representatives. The 2LDP was extended by including a lookahead (2LDP-LA) that influences the column alignments by estimating the costs of the rest of the warping. This lookahead was refined by introducing the aligned lookahead (2LDP-ALA). While 2LDP-LA uses a simple backward run of dynamic programming as lookahead, with 2LDP-ALA scores of a previous top-down and bottom-up run are used to align the paths computed during a backward run. The last extension was also applied to C-TSDP (C-TSDP-ALA). Additionally, several modifications to 2LDP have been proposed that simplify the complexity of the algorithm by reducing the number of candidates considered for each column warping or restrain the possible warpings by applying a strip-restriction (2LDP-S).

For the evaluation two databases were used. A subset of the AR-Face database was modified by rotating the images by up to 45 degrees to raise the difficulty. As second database the pose subset of the CMU-PIE database was used. The experiments on both databases showed that the scores achieved by 2LDP are significantly improved by using the lookahead. Replacing the lookahead by the aligned lookahead improves the score further. The same effect was observed for C-TSDP, where the scores are improved by applying the aligned lookahead. Unfortunately, the better scores do not always lead to better error rates. While this is still the case for the Rotated-45 AR-Face database, on the CMU-PIE database a different effect was observed. Here, mostly the between-class warpings profit from the improved scores leading to a deteriorated error rate.

The comparison of 2LDP with its modifications revealed that only with no lookahead the

additional candidates considered by 2LDP lead to improved warpings. Otherwise, the column alignments are strongly influenced by the lookahead increasing the similarity of the different candidates. For the aligned lookahead already C-TSDP as baseline achieves similar results. Also the conclusion regarding 2LDP-S is depending on the lookahead. While the 2LDP error rate is improved by the strip-restriction in case no lookahead is used, the positive effect is lost when the lookahead is applied. For the former the restriction provides 2LDP with a weak orientation for the warpings. However, the information provided by the lookahead is much more helpful such that the strip-restriction is rendered obsolete.

Compared to the state-of-the-art warping methods such as TSDP, Pseudo Two-Dimensional Warping with First-Order Strip Extension (P2DW-FOSE) and Sequential Tree-Reweighted Message Passing (CTRW-S), the introduced algorithms are competitive, but the error rates are not improved. Especially for CTRW-S, the only model that also computes the warping within complete Sakoe constraints, the error rates are similar. On the Rotated-45 AR-Face database the best result of 2LDP was an error rate of 20.00%, for C-TSDP the best result was an error rate of 20.95%. This is close to the 18.10% error rate achieved by CTRW-S, however the best result was obtained using TSDP leading to an error rate of 13.30%. On the CMU-PIE database CTRW-S leads to the best error rate of 3.43%. Again, C-TSDP and 2LDP achieve a similar result with error rates of 3.68% and 4.66% respectively.

As we have seen, with the best lookahead the optimization performed by 2LDP on the global level leads only to small improvements. In order to justify the computational overhead of 2LDP compared to its modifications, the variability of the candidates needs to be increased. Judging by the strong impact of the lookahead on the results, further extensions might be worth looking into. For instance, when the columns are aligned pixel by pixel, the lookahead is considered only in the horizontal dimension. Here, an additional vertical lookahead could be used.

# Appendix A

## Appendix

### A.1 Software

The novel methods introduced in Chapter 4 have been implemented in C and integrated into the W2D-Software. The latter is a software package that originated in [Gollan 03] and serves as general warping framework. The new parameters are as follows.

- `-M2LDP` to select 2LDP as warping model.
- `-MCTSDP` to select C-TSDP as warping model.
- `-c 0` to set the representative / starting column to 0.
- `-l 1.0` to set the lookahead weight  $\gamma = 1.0$ .
- `-q` to enable the aligned lookahead.

Below are a few command-line examples to show how the new models can be used within the software.

- 2LDP-ALA experiment for one pose on the CMU-PIE (cf. Table 7.5).  

```
./W2D
usift-pose27-64x64-pca30-patchnorm.ff
usift-pose37-64x64-pca30-patchnorm.ff
-M2LDP -c32 -l1.0 -q -i -Q2
-Dfeature:absrec5x5:1,penalty:pen4:0.001 -k1.1
```
- C-TSDP experiment for one pose on the CMU-PIE with the first column as starting column (cf. Table 7.3).  

```
./W2D
usift-pose27-64x64-pca30-patchnorm.ff
usift-pose37-64x64-pca30-patchnorm.ff
-MCTSDP -c0 -l1.0 -i -Q2
-Dfeature:absrec5x5:1,penalty:pen4:0.001 -k1.1
```



## List of Figures

3.1	Illustration of two-dimensional warping. . . . .	8
3.2	Horizontal Sakoe constraints. . . . .	10
3.3	Vertical Sakoe constraints. . . . .	10
3.4	Rotation of at most 90 degrees clockwise. . . . .	11
3.5	Rotation of at most 90 degrees counter-clockwise. . . . .	11
3.6	Illustration of ZOW. . . . .	12
3.7	Illustration of P2DW . . . . .	14
3.8	Illustration of P2DW-FOSE . . . . .	15
3.9	Illustration of TSDP. . . . .	16
4.1	Illustration of C-TSDP. . . . .	20
4.2	Illustration of 2LDP. . . . .	22
4.3	Global level of 2LDP. . . . .	24
4.4	Initialization of 2LDP. . . . .	25
4.5	Column-level of 2LDP. . . . .	26
4.6	Global level of the 2LDP-U. . . . .	27
4.7	Global level of the 2LDP-LB. . . . .	28
4.8	Illustration of 2LDP-S during initialization. . . . .	30
4.9	Illustration of the lookahead. . . . .	31
4.10	Illustration of the aligned lookahead. . . . .	33
6.1	The Rotated-45 AR-Face database. . . . .	42
6.2	The CMU-PIE pose database. . . . .	42
7.1	Rotated-45 AR-Face: Results for ZOW with different warpranges $\Delta$ . . . . .	44
7.2	ZOW examples for images with no, 25 degrees and 45 degrees rotation. . . . .	46
7.3	Rotated-45 AR-Face: Results of the P2DW-FOSE with different strip-width $\Delta$ . . . . .	46
7.4	P2DW-FOSE examples for images with no, 25 degrees and 45 degrees rotation. . . . .	47
7.5	TSDP examples for images with no, 25 degrees and 45 degrees rotation. . . . .	49
7.6	Rotated-45 AR-Face: Results of the CTRW-S. . . . .	49
7.7	CTRW-S examples for images with no, 25 degrees and 45 degrees rotation. . . . .	50
7.8	Rotated-45 AR-Face: Comparison of C-TSDP-LA and C-TSDP-ALA. . . . .	52
7.9	Rotated-45 AR-Face: Comparison of C-TSDP-ALA and CTRW-S. . . . .	53
7.10	CMU-PIE: Comparison of C-TSDP and C-TSDP-ALA. . . . .	53

7.11 C-TSDP examples for images with no, 25 degrees and 45 degrees rotation. . .	54
7.12 Rotated-45 AR-Face: Comparison of 2LDP and 2LDP-LA. . . . .	56
7.13 Rotated-45 AR-Face: Comparison of 2LDP-LA and 2LDP-ALA. . . . .	56
7.14 2LDP examples for images with no, 25 degrees and 45 degrees rotation. . . .	57
7.15 2LDP-ALA examples for images with no, 25 degrees and 45 degrees rotation.	58
7.16 Rotated-45 AR-Face: Comparison of C-TSDP and 2LDP with lookahead. . .	59
7.17 CMU-PIE: Comparison of C-TSDP and 2LDP-LA. . . . .	60
7.18 Rotated-45 AR-Face: Comparison of 2LDP-ALA and CTRW-S. . . . .	60
7.19 Rotated-45 AR-Face: Comparison of 2LDP and 2LDP-U. . . . .	62
7.20 Rotated-45 AR-Face: Comparison of 2LDP and 2LDP-LB. . . . .	63
7.21 Rotated-45 AR-Face: Results for 2LDP-U-ALA with different values for $\gamma$ . .	64
7.22 Rotated-45 AR-Face: Results of 2LDP-ALA with different representatives. .	65
7.23 Rotated-45 AR-Face: The effect of using nearest neighbor pruning. . . . .	66

## List of Tables

4.1	Sakoe constraint enforcement of the different warping algorithms. . . . .	36
4.2	Complexities of the warping methods. . . . .	37
7.1	Rotated-45 AR-Face: Effect of absolute warping penalty. . . . .	45
7.2	Rotated-45 AR-Face: Results of the C-TSDP compared to the TSDP. . . . .	51
7.3	CMU-PIE: Results of the C-TSDP compared to the TSDP. . . . .	53
7.4	Rotated-45 AR-Face: Results of 2LDP with and without lookaheads . . . . .	55
7.5	CMU-PIE: Results of 2LDP with and without lookahead. . . . .	56
7.6	Rotated-45 AR-Face: Results of 2LDP compared to 2LDP-U. . . . .	61
7.7	Rotated-45 AR-Face: Results of 2LDP with the local-best strategy. . . . .	62
7.8	Rotated-45 AR-Face: Results of the 2LDP with and without strip-restriction. . . . .	63
7.9	CMU-PIE: Results of the 2LDP compared to the 2LDP-S. . . . .	64
7.10	Rotated-45 AR-Face: Overview of the results. . . . .	67
7.11	CMU-PIE: Overview of the results. . . . .	68



## Bibliography

- [Ahonen & Hadid<sup>+</sup> 04] T. Ahonen, A. Hadid, M. Pietikainen: Face recognition with local binary patterns. In *European Conference on Computer Vision (ECCV)*, pp. 469–481. Springer, May 2004.
- [Arashloo & Kittler 09] S. Arashloo, J. Kittler: Hierarchical image matching for pose-invariant face recognition. In *British Machine Vision Conference (BMVC)*, September 2009.
- [Belhumeur & Hespanha<sup>+</sup> 97] P. Belhumeur, J. Hespanha, D. Kriegman: Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 711–720, 1997.
- [Berretti & Del Bimbo<sup>+</sup> 10] S. Berretti, A. Del Bimbo, P. Pala: 3D Face Recognition Using Isogeodesic Stripes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol., pp. 2162–2177, 2010.
- [Bicego & Lagorio<sup>+</sup> 06] M. Bicego, A. Lagorio, E. Grosso, M. Tistarelli: On the use of SIFT features for face authentication. pp. 35–35, June 2006.
- [Blanz & Vetter 03] V. Blanz, T. Vetter: Face recognition based on fitting a 3D morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol., pp. 1063–1074, 2003.
- [Dreuw & Steingrube<sup>+</sup> 09] P. Dreuw, P. Steingrube, H. Hanselmann, H. Ney: Surf-face: Face recognition under viewpoint consistency constraints. In *British Machine Vision Conference (BMVC)*, Sept. 2009.
- [Ekenel & Stiefelhagen 06] H. Ekenel, R. Stiefelhagen: Analysis of local appearance-based face recognition: Effects of feature selection and feature normalization. June 2006.
- [Gass & Dreuw<sup>+</sup> 10] T. Gass, P. Dreuw, H. Ney: Constrained Energy Minimisation for Matching-Based Image Recognition. In *International Conference on Pattern Recognition*, accepted for publication, Istanbul, Turkey, Aug. 2010.
- [Gass & Pishchulin<sup>+</sup> 11] T. Gass, L. Pishchulin, P. Dreuw, H. Ney: Warp that Smile on your Face: Optimal and Smooth Deformations for Face Recognition. In *IEEE International*

- Conference Automatic Face and Gesture Recognition*, Santa Barbara, CA, USA, March 2011.
- [Gollan 03] C. Gollan: Nichtlineare Verformungsmodelle für die Bilderkennung. Diploma Thesis, Sept. 2003.
- [Gross & Shi<sup>+</sup> 01] R. Gross, J. Shi, J. Cohn: Quo vadis face recognition. In *Third Workshop on Empirical Evaluation Methods in Computer Vision*, Vol. 2, pp. 7–2, 2001.
- [Hua & Akbarzadeh 09] G. Hua, A. Akbarzadeh: A robust elastic and partial matching metric for face recognition. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 2082–2089, Sept. 2009.
- [Jähne 02] B. Jähne: *Digitale Bildverarbeitung*. Vol. 5th Edition. Springer-Verlag, 2002.
- [Ke & Sukthankar 04] Y. Ke, R. Sukthankar: PCA-SIFT: A more distinctive representation for local image descriptors. In *IEEE Computer Vision and Pattern Recognition Workshop (CVPRW)*, 506–513, June 2004.
- [Keysers & Deselaers<sup>+</sup> 07] D. Keysers, T. Deselaers, C. Gollan, H. Ney: Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol., pp. 1422–1435, 2007.
- [Keysers & Unger 03] D. Keysers, W. Unger: Elastic image matching is NP-complete. *Pattern Recognition Letters*, Vol. 24, No. 1-3, pp. 445–453, 2003.
- [Kolmogorov 06] V. Kolmogorov: Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol., pp. 1568–1583, 2006.
- [Kuo & Agazzi 94] S. Kuo, O. Agazzi: Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 16, No. 8, pp. 842–848, 1994.
- [Lowe 99] D. Lowe: Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, Sept. 1999.
- [M. Wainwright 02] A.W. M. Wainwright, T. Jaakkola: MAP estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. *IEEE Transactions on Information Theory*, Vol. 51, pp. 3697–3717, 2002.
- [Martinez & Benavente 98] A. Martinez, R. Benavente: The AR face database. Technical report, CVC Technical report, 1998.
- [Martinez & Kak 01] A. Martinez, A. Kak: Pca versus lda. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 2, pp. 228–233, 2001.

- [Mottl & Kopylov<sup>+</sup> 02] V. Mottl, A. Kopylov, A. Kostin, A. Yermakov, J. Kittler: Elastic transformation of the image pixel grid for similarity based face identification. *Pattern Recognition*, Vol. 3, 2002.
- [Ney & Dreuw<sup>+</sup> 10] H. Ney, P. Dreuw, T. Gass, L. Pishchulin: Image Recognition and 2D Warping. Lecture Notes. RWTH Aachen University, 2010.
- [Pinto & DiCarlo<sup>+</sup> 09] N. Pinto, J. DiCarlo, D. Cox: How far can you get with a modern face recognition test set using only simple features? In *IEEE Computer Vision and Pattern Recognition Workshop (CVPRW)*, June 2009.
- [Pishchulin 10] L. Pishchulin: Matching Algorithms for Image Recognition. Master's thesis, RWTH Aachen University, Aachen, Germany, Jan. 2010.
- [Pishchulin & Gass<sup>+</sup> 11] L. Pishchulin, T. Gass, P. Dreuw, H. Ney: The Fast and the Flexible: Extended Pseudo Two-Dimensional Warping for Face Recognition. In *Iberian Conference on Pattern Recognition and Image Analysis*, Gran Canaria, Spain, June 2011.
- [Scheenstra & Ruifrok<sup>+</sup> 05] A. Scheenstra, A. Ruifrok, R. Veltkamp: A survey of 3D face recognition methods. In *Audio-and Video-Based Biometric Person Authentication (AVBPA)*, pp. 891–899, July 2005.
- [Shakhnarovich & Moghaddam 04] G. Shakhnarovich, B. Moghaddam: Face recognition in subspaces. *Handbook of Face Recognition*, Vol., pp. 141–168, 2004.
- [Sim & Baker<sup>+</sup> 02] T. Sim, S. Baker, M. Bsat: The CMU Pose, Illumination, and Expression (PIE) Database. In *IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, May 2002.
- [Tan & Chen 05] K. Tan, S. Chen: Adaptively weighted sub-pattern PCA for face recognition. *Neurocomputing*, Vol. 64, pp. 505–511, 2005.
- [Tan & Chen<sup>+</sup> 06] X. Tan, S. Chen, Z. Zhou, F. Zhang: Face recognition from a single image per person: A survey. *Pattern Recognition*, Vol. 39, No. 9, pp. 1725–1745, 2006.
- [Turk & Pentland 91] M. Turk, A. Pentland: Face recognition using eigenfaces. In *IEEE Computer Vision and Pattern Recognition Workshop (CVPRW)*, pp. 586–591, June 1991.
- [Uchida & Sakoe 98] S. Uchida, H. Sakoe: A monotonic and continuous two-dimensional warping based on dynamic programming. In *International Conference on Pattern Recognition (ICPR)*, Vol. 1, pp. 521–524, Aug. 1998.
- [Wiskott & Fellous<sup>+</sup> 97] L. Wiskott, J. Fellous, N. Kuiger, C. Von der Malsburg: Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 775–779, 1997.

- [Xu & Li<sup>+</sup> 09] C. Xu, S. Li, T. Tan, L. Quan: Automatic 3D face recognition from depth and intensity Gabor features. *Pattern Recognition*, Vol. 42, No. 9, pp. 1895–1905, 2009.
- [Zhang & Gao<sup>+</sup> 08] X. Zhang, Y. Gao, M. Leung: Recognizing rotated faces from frontal and side views: An approach toward effective use of mugshot databases. *IEEE Transactions on Information Forensics and Security*, Vol. 3, No. 4, pp. 684–697, 2008.
- [Zhang & Samaras 06] L. Zhang, D. Samaras: Face recognition from a single training image under arbitrary unknown lighting using spherical harmonics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 3, pp. 351–363, 2006.
- [Zhao & Chellappa<sup>+</sup> 03] W. Zhao, R. Chellappa, P. Phillips, A. Rosenfeld: Face recognition: A literature survey. *Acm Computing Surveys (CSUR)*, Vol. 35, No. 4, pp. 399–458, 2003.