Low-rank Based Algorithms for Rectification, Repetition Detection and De-noising in Urban Images

Juan Liu
Graduate Center, City University of New York
Low-rank Based Algorithms for Rectification, Repetition Detection and De-noising in Urban Images

by

Juan Liu

A dissertation submitted to the Graduate Faculty in

Computer Science

in partial fulfillment of the requirements for

the degree of Doctor of Philosophy

THE CITY UNIVERSITY OF NEW YORK

2015
This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy

Dr. Ioannis Stamos

Date Chair of Examining Committee

Dr. Robert Haralick

Date Executive Officer

Dr. Emmanouil Z. Psarakis, University of Patras, Greece
Dr. Yingli Tian, City College, CUNY
Dr. Zhigang Zhu, City College, CUNY
Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK
ABSTRACT

Low-rank Based Algorithms for Rectification, Repetition Detection and De-noising in Urban Images

by

Juan Liu

Advisor: Dr. Ioannis Stamos

In this thesis, we aim to solve the problem of automatic image rectification and repeated patterns detection on 2D urban images, using novel low-rank based techniques. Repeated patterns (such as windows, tiles, balconies and doors) are prominent and significant features in urban scenes.

Detection of the periodic structures is useful in many applications such as photorealistic 3D reconstruction, 2D-to-3D alignment, façade parsing, city modeling, classification, navigation, visualization in 3D map environments, shape completion, cinematography and 3D games. However both of the image rectification and repeated patterns detection problems are challenging due to scene occlusions, varying illumination, pose variation and sensor noise. Therefore, detection of these repeated patterns becomes very important for city scene analysis.

Given a 2D image of urban scene, we automatically rectify a façade image and extract façade textures first. Based on the rectified façade texture, we exploit novel algorithms that extract repeated patterns by using Kronecker product based modeling that is based on a solid theoretical foundation. We have tested our algorithms in a large set of images, which includes building façades from Paris, Hong Kong and New York.
Acknowledgments

My deepest gratitude is to my advisor, Prof. Ioannis Stamos. I have been amazingly fortunate to have such a great advisor who gave me the freedom to explore on my own and at the same time the guidance to recover when my steps faltered. He taught me how to question thoughts and express ideas. His patience and support helped me overcome many crisis situations and finish this dissertation.

I will forever be grateful to Prof. Emmanouil Z. Psarakis for his scientific advice and knowledge and many insightful discussions and suggestions. He is my primary resource for getting my science questions answered and was instrumental in helping me crank out this thesis in the past few years. I am thankful to Prof. Yang Feng, who, as a good friend, was always willing to help me and offer his best suggestions.

Many thanks to Prof. Yingli Tian and Prof. Zhigang Zhu for being members of my thesis committee and for their interest and guidance. Thanks to my labmates Allan Zelener, Tom Flynn, Sam Freeman and Agis Mesolongitis. We spent a lot of time together, constantly helping each other. I sincerely thank Prof. Robert Haralick and Prof. Susan Epstein for their help and encouragement in my darkest moments. I am also grateful to my friends who have helped me stay sane through these difficult years. Their support and care helped me get over setbacks and stay focused on my graduate study.

Most importantly, none of this would have been possible without the love and support from my beloved parents, Dajian Liu and Guiying Wu, and my little brother Jun Liu. My family, to whom this dissertation is dedicated, have been a constant source of love, concern, support and strength all these years. I love you so much.

Finally, my special thanks go to my husband, Hongjun Zheng, who was always there loving me, cheering me up and standing by me through the good times and bad.
Table of Contents

1 Introduction 1
  1.1 Big Picture: Computer Vision and Image Processing 1
  1.2 Problem Definition 3
  1.3 Contribution 4

2 Background 6
  2.1 Geometric Primitives 6
    2.1.1 2D Images 6
    2.1.2 3D to 2D Projection 7
    2.1.3 The 2D Projective Transformation 8
  2.2 Transformation Estimation Methods 9
  2.3 Façade Analysis and Repetitive Patterns Detection 11
    2.3.1 Low-rank Based Façade Analysis 12
    2.3.2 Heuristic Segmentation 13
    2.3.3 Façade Parsing via Shape Grammars 13
    2.3.4 Symmetry Detection 15
    2.3.5 Machine Learning Methods 17
    2.3.6 Interactive Façade Modeling Methods 17
  2.4 3D Reconstruction 18

3 Rectifying 2D Façade Images 20
# 3.1 Algorithm Overview ......................................................... 21
# 3.2 Detecting Features .......................................................... 23
## 3.2.1 Vanishing Points ......................................................... 23
## 3.2.2 Harris Corners ............................................................ 23
# 3.3 Automatically Selecting A Façade Region ............................ 24
# 3.4 The Urban Low-rank Algorithm for Façade Image Rectification 27
## 3.4.1 Problem Formulation ..................................................... 28
## 3.4.2 Solve the Problem via Alternating Direction Methods ....... 30
# 3.5 Experiments and Analysis ................................................... 32

# 4 Kronecker Product Model: Theory ........................................ 37
## 4.1 Introduction ................................................................. 37
### 4.1.1 Motivation ............................................................. 37
### 4.1.2 The Kronecker Product ............................................... 38
## 4.2 Modeling a Building Façade via Kronecker Products .......... 39
### 4.2.1 Ideal Façade Modeling ................................................ 39
## 4.3 Recovering the Ideal Façade Model .................................... 42
## 4.4 Discussion ................................................................. 45

# 5 Applying the Kronecker Product Model to Repeated Patterns Detection 47
## 5.1 Introduction .................................................................. 47
## 5.2 Algorithm ................................................................. 48
### 5.2.1 Estimating the Spatial Periods of the Patterns ............... 48
### 5.2.2 Estimating $K$ by Unbiased Estimator of the Degrees of Freedom ......................................................... 54
### 5.2.3 Estimating Matrices $M_k, k = 1, 2, \ldots, K^*$. .......... 59
### 5.2.4 Computing Patterns and Weighting Factors ............... 59
### 5.2.5 Patterns Refinement .................................................. 59
## 5.3 Experiments and Discussion ........................................... 64
## 5.4 Summary ................................................................. 64
6 Conclusion

6.1 Novelty and Contributions of Our Methods .............................................. 66

6.2 Limitations and Future Direction .............................................................. 67

6.2.1 Automatic Texture Selection and Façade Segmentation ...................... 67

6.2.2 Nested Patterns ..................................................................................... 68

6.2.3 Types of Patterns .................................................................................. 70

6.2.4 3D Point Clouds ................................................................................. 70

Bibliography .................................................................................................... 70
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Speed Comparison of TILT and our urban low-rank algorithm, with branch-and-bound initialization and manually selected texture</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Performance comparison between the branch-and-bound initialization and vanishing points based transform initialization. The vanishing points initialization results in significant increase in success rate without a penalty in speed. The experiments are done on a set of 306 façade images</td>
<td>34</td>
</tr>
</tbody>
</table>
List of Figures

2.1 A gray-level 2D image formulation. (a) An image $I$ can be represented as a two-dimensional surface. (b) The image $I$ can be represented as a two-dimensional table of integers which denotes the intensity. (c) The visualization of the image $I$ in (a). [Ma, 2004] 7

2.2 A projective rectification of the plane. (a) The given four points on the scene plane, and (b) the rectified plane image. This method does not require knowledge of any of the cameras parameters or the pose of the plane. [Hartley and Zisserman, 2003]. 9

3.1 An image rectification example. (a) The original image, on which the red box is the input of our algorithm and the green quadrilateral indicates the transformation. (b) The rectified image texture. (c) The low-rank component detected by our algorithm, where $\text{rank} = 6$. 22

3.2 Texture selection procedure. (a) The blue lines denote the directions of the major vanishing points. (b) Hypothetical lines drawn from vanishing points towards image plane. (c) 10 by 10 blocks divided along vanishing points directions. (d) Harris corners features extracted for block selection. (e) The yellow box localizes a raw texture selected by the algorithm. (f) The green quadrilateral is the isolated yellow area in (e). (g) A rectified façade image through a pure vanishing points algorithm. (h) The largest rectangular box that fits inside the selected region, to be used for initialization. (i) A rectified façade by the urban low-rank algorithm with the red box in figure (e) as input texture. 26
3.3 Representative examples of rectified urban building façade structures. Due to occlusions and noise, their corresponding matrices are full rank, even though there is an underlying low-rank component that we wish to reveal.

3.4 A representative low-rank edge: (a) the deformed edge with high rank, (b) rank $= 1$.

[Zhang et al., 2010]

3.5 Comparison between TILT and our urban low-rank algorithm. (a) The red box is the input texture, and the green box denotes the output of our algorithm, (b) the rectified façade by our urban low-rank algorithm, (c) the red box is the input for TILT, and green quadrilateral denotes the transformation $\tau$ recovered by TILT, and (d) the final rectification result from TILT, where TILT failed to rectify this image.

3.6 All the initial input images that correspond to different pair of rotation angle and skew parameter. The rotation angle range from 0 to 45, with a step of 5 degree, while the skew parameter is evaluated from 0.0 to 0.9, with a gap of 0.05.

3.7 Range of convergence for affine transformation. $x$-axis: rotation angle $\theta$. $y$-axis: skew parameter $t$. The entire region on and below the line indicates success in all trials while the region above the line indicates failure in all trials.

4.1 A synthetic urban building façade based on the model of Eq.(4.3) with weighting coefficients $\lambda_1 = 40$, $\lambda_2 = 160$, $\lambda_3 = 80$ and $\lambda_4 = 255$.

5.1 A real building façade (top), its optimal modeling of rank 4 (middle), obtained from the solution of the optimization problem of Eq. (4.10), and its optimal modeling of rank 4 (bottom), obtained from the minimization of the cost function of Eq. (4.17) with matrices $M_k$ predefined (please see text).

5.2 Estimation of the spatial periods of façade shown in Figure 5.1 (left). (a) Cross-Correlation sequences used for the estimation of $N_v = 90$ pixels and (b) estimation of $N_h = 56$ pixels. Please enlarge the image to see the coordinates. Distance between the adjacent peaks provides the period information.
5.3 This figure shows the rank estimation result for a real façade building image shown in Figure 5.1 (top). (a) Illustration of the estimation function in Eq. (5.20), where the global minima comes up at index 4. The corresponding index indicates the desired rank for the re-arranged matrix $F$ shown in Figure 5.4. Please see Algorithm 6 for the estimation process. (b) The enlarged figure area inside the green box in (a), which shows the global minima in a clearer way. (c) Illustration of the first term of function in Eq. (5.21). (d) Illustration of the degrees of freedom function in Eq. (5.19).

5.4 Vectorization of all the $3 \times 10$ blocks in Figure 5.7 (b). This matrix is of full rank.

5.5 The low-rank component of matrix in Figure 5.4, where $K = 4$. The rank is estimated by Algorithm 6.

5.6 Refining the representative pattern for each group.

5.7 (a) Partitioning the façade into blocks by using the spatial periods estimated in Figure 5.2, where the green lines shows the partition blocks’ boundary. (b) Isolating the corresponding $3 \times 10$ blocks.

5.8 (a) Reshaping each row vector of matrix in Figure 5.5 to partition blocks. (b) Reconstructing façade image by pasting all blocks together.

5.9 Grouping. Each color represents one group, and all patterns in a group have the same structure.

5.10 An overview of the repeated pattern detection procedure via Kronecker Product model. (a) Input image. (b) Partition grid showing the periods estimated by Algorithm 5, with all partition blocks colored randomly. (c) Grouped blocks generated by Algorithms 4 and 7, with each group having the same color. (d) Low-rank component generated by Algorithms 4 - 5 in Sections 5.2.1.1 - 5.2.1. (e) Estimated 1-0 repeated patterns by refining detection results shown in (d).

5.11 From left to right: follow the order of Figure 5.10. The third row is a failure case of method presented in [Zhao and Quan, 2011] (large tree occlusion), but our method can successfully detect the repeated patterns as shown in the last column. The fifth row shows detected patterns for the example in Figure 5.1.
5.12 From left to right: follow the order of Figure 5.10. First eight rows show success cases (robustness to occlusions and different architectural styles). Last row is a failure case due to the photometric variation and inability to model via a Kronecker product model.

6.1 A façade example that contains nested patterns. (a) The input image where the red box shows a nested pattern that contains three single windows. (b) By adopting nested patterns, the façade can be partitioned into repeated textures that contains multiple indivisible structures like doors and windows.

6.2 The Cross-Correlation function along horizontal direction for the façade in Figure 6.1. There actually are at least two types of repetition periods, denoted by the higher peaks and low peaks respectively. Top: in the current model, the distance between two adjacent peaks of the Cross-Correlation function is computed as the period. Bottom: we can find a strategy to isolate the higher peaks for the nested patterns. The façade can be partitioned into repeated textures that contains multiple indivisible structures as shown in Figure 6.1.

6.3 (a) A 3D point cloud of the city scene from the Wright State Ottawa dataset. (b) A 3D point cloud of a car. [Zelener et al., 2014]
To my beloved family.
Chapter 1

Introduction

1.1 Big Picture: Computer Vision and Image Processing

In the real world, humans and animals perceive the three-dimensional structure of the world with apparent cases. Imagining how vivid the three-dimensional perception is when you look at an arrangement of flowers, you can easily tell the shape and translucency of each petal and effortlessly segment each flower from the background. Looking at a 2D portrait image of soccer stars, you can easily count and name all of the soccer stars and even guess at their emotions from their facial appearance [Szeliski, 2010].

In computer vision, we seek to recover some unknowns given insufficient information to fully specify the solution, which is called an inverse problem. More specifically, we are trying to describe the world that we see in one or more images and to reconstruct its properties, such as shapes, illuminations, and colors. Computer vision algorithms need to exploit physics-based and probabilistic models to disambiguate between potential solutions.

In the past decades, researchers in computer vision have been developing mathematical techniques for recovering the three-dimensional shape and appearance of objects in images. We now have reliable techniques for accurately computing a partial 3D model of an environment from thousands of partially overlapping photographs [Snavely et al., 2006]. Given enough set of views of a particular façade, we can create accurate 3D surface models by using stereo matching [Goesele...
CHAPTER 1. INTRODUCTION

We can detect and track a person or vehicle moving against a complex background [Sidenbladh et al., 2000]. By using a combination of face and hair detection and recognition, we can even attempt to find and name all of the people in a photograph [Sivic et al., 2006]. While the techniques and software are developing, people are creating better hardware and tools for data acquisition, such as high resolution 2D digital cameras and 3D spot laser scanner [Stamos and Allen, 2001]. However, despite all of these advances, having a computer to interpret an image at the same level as a two-year old remains extremely difficult.

Computer vision is being used today in a wide variety of real-world applications, which include optical character recognition (OCR), retail, medical imaging, motion capture, surveillance, fingerprint recognition and biometrics, stitching, 3D modeling, and stabilization, face detection and visual authentication, etc. 3D reconstruction is the creation of three-dimensional models from one or multiple images, such as in the street view in Google Maps system and the 3D Apple Maps. It is the reverse process of obtaining 2D images from 3D scenes. 3D scene modeling has been one of the longest studied problems in computer vision. Recently, the development of reliable image-based modeling techniques has spurred renewed interest in this area along with the prevalence of digital camera and 3D computer games.

The essence of an image is a projection from a 3D scene onto a 2D image plane, during which process the depth is lost. The 3D point corresponding to a specific image point is constrained to be on the line of sight. The task of converting 2D images into a 3D model in urban scene consists of a series of processing steps: camera calibration, registration, façade structure analysis and geometry reconstruction.

Recovering clean façade structure is essential for reconstructing a complete 3D model, as most of the acquired 2D images are perspectively distorted and occluded. In this thesis, we aim to solve two fundamental problems: automatic image rectification, façade texture detection and repetitive patterns detection on 2D urban images.
1.2 Problem Definition

Urban scenes contain rich periodic or near-periodic structures, such as windows, doors, and other architectural features. Detection of the periodic structures is useful in many applications such as photorealistic 3D reconstruction, 2D-to-3D alignment, façade parsing, city modeling, classification, navigation, visualization in 3D map environments, shape completion, cinematography and 3D games. However it is a challenging task due to scene occlusion, varying illumination, pose variation and sensor noise.

In recent years, repeated patterns or periodic structures detection has received significant attention in both 2D images [Zhao et al., 2010], [Teboul et al., 2011b] and 3D point clouds [Friedman and Stamos, 2013], [Shen et al., 2011]. Repeated patterns are usually hypothesized from the matching of local image features. They can be modeled as a set of sparse repeated features [Schindler et al., 2008a] in which the crystallographic group theory [Liu et al., 2004a] was employed. The work of [Wu et al., 2010a] maximizes local symmetries and separates different repetition groups via evaluation of the local repetition quality conditionally for different repetition intervals.

The work of [Muller et al., 2007a] proposes an approach to detect symmetric structures in a rectified fronto-façade and to reconstruct a 3D geometric model. The work of [Yang et al., 2012a] describes a method for periodic structure detection upon the pixel-classification results of a rectified façade. The first step of all façade parsing algorithms (see [Yang et al., 2012a] for an example) is the detection and rectification of individual façade structures. Shape grammars have also been used for 2D façade parsing [Teboul et al., 2011b]. Another similar grammar-based approaches is presented by [Barinova et al., 2010].

All the methods mentioned above require as pre-processing image rectification. To solve this problem, low-rank methods attracted a lot of attention in recent years. Zhang and colleagues [Zhang et al., 2010] proposed a low-rank algorithm for rectifying an image starting with a manually selected representative texture that is followed by a branch-and-bound initialization scheme. A similar work was proposed by [Gandy et al., 2011] in which the rank value $N$ is assumed known. Another method for recovering both the low-rank component and the sparse error is presented in
In addition, [Liu et al., 2013a] describes a low-rank based method that detects repetitive patterns in 2D images for the application of shape completion.

In this thesis, we aim to develop low-rank algorithms for solving three major problems in urban scene analysis: automatically rectifying façade images, extracting façade textures and detecting repetitive patterns on rectified façade textures.

1.3 Contribution

We contribute novel low-rank based methods with theoretical justification and experimental support:

- Image rectification algorithms (Chapter 3). These methods aim to rectify the image precisely and automatically by combining the traditional feature-based methods and low-rank techniques. This approach takes advantages of the information offered by both vanishing points and low-rank invariant features, but avoids their weakness.

- Automatic façade texture extraction algorithms (Chapter 3). These methods detect the façade regions in 2D urban images. By combining the image rectification algorithm, frontal façades are generated for a variety of fundamental tasks, such as façade parsing, repetitive patterns detection and 3D reconstruction. We adopt Harris corners and vanishing lines in these algorithms.

- Repetitive patterns detection algorithms on frontal façades (Chapters 4 and 5). We are the first to propose the novel Kronecker Product model for modeling a frontal façade structure globally in an efficient and effective way. The Kronecker Product model integrates low-rank textures, robust PCA and the Kronecker Product, and has been proved based on a solid theoretical foundation. The input of this method is a frontal façade texture, which is the output of the approach described in Chapter 3. This model not only recovers repeated patterns, but also removes noise and occlusions. The algorithms are general and can be applied to a wide variation of façade structures. The fact that we are utilizing the low-rank
part of the rearranged input façade image allows us to handle the problem of occlusion, shadows and illumination variation.
Chapter 2

Background

2.1 Geometric Primitives

Before intelligently analyzing and manipulating images, we need to understand the geometry of a scene and the image formation process that produced a particular image.

2.1.1 2D Images

An image, $I$, usually is represented by a two-dimensional array, in which each entry represents the brightness. The higher a matrix entry is, the brighter this corresponding pixel is. More specifically, it can be interpreted a map defined on a compact region $\Omega$ of a two-dimensional surface, where the values range in the positive real numbers $[0, 255]$. In the case of digital cameras, $\Omega$ is a planar and rectangular region occupied by the CCD sensor. A 2D point (pixel coordinates in an image) can be denoted using a pair of values $(x, y) \in \mathbb{R}^2$. Before describing the image formation process, we must specify the value of $I(x, y)$ at each point $(x, y)$ in $\Omega$. Such a value $I(x, y)$ is typically called image intensity or brightness. Thus $I$ can be defined as a function:

$$I: \Omega \subset \mathbb{R}^2 \to \mathbb{R}_+; \quad (x, y) \mapsto I(x, y). \quad (2.1)$$

Such an image can be visualized by using the graph of $I$ as in the example in Figure 2.1(a).
Both the domain Ω and the range \( \mathbb{R}_+ \) are discretized. For example, in a 2D graylevel image, we may take \( \Omega = [1,1280] \times [1,960] \subset \mathbb{Z}^2 \) and \( \mathbb{R}_+ \) can be approximated by an interval of integers \([0,255] \subset \mathbb{Z}_+\). Such an image can be represented by a two-dimensional matrix of numbers as shown in Figure 2.1(b). A picture of the same image \( I \) described in Figure 2.1(a) and Figure 2.1(b) is illustrated in Figure 2.1(c). Although the picture seems more informative, it is merely a different representation and contains exactly the same information.

### 2.1.2 3D to 2D Projection

Now that we know how to represent a 2D image, we need to specify how 3D scene is projected onto the image plane. A 2D image essentially is the projection of the 3D scene. Given a generic 3D point \( p \), with coordinates \( X_0 = [X_0, Y_0, Z_0]^T \in \mathbb{R}^3 \), the coordinates \( X = [X, Y, Z]^T \) of the same point \( p \) relative to the camera frame are given by a transformation \( g = (R, T) \) of \( X_0 \):

\[
X = RX_0 + T \in \mathbb{R}^3,
\]

where \( R \) and \( T \) denote a rotation and translation respectively.

Considering the frontal pinhole camera model, the point \( X \) is projected onto the image plane at the point \[Ma, 2004\]
CHAPTER 2. BACKGROUND

\[ \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}. \quad (2.3) \]

In homogeneous coordinates, this relationship can be written as

\[
Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (2.4)
\]

2.1.3 The 2D Projective Transformation

The most commonly used model for 3D real world is perspective transformation, since it more accurately models the behavior of real cameras. When we look at a 2D image, we see squares that are not squares, and circles that are not circles. The transformation that maps these planar objects onto the picture is an example of a projective transformation, in which most of the properties of geometry, such as shape, angle, distance and ratios of distances, are not preserved. People are seeking invariant features by a projective transformations. One such preserved property is straightness. The perspective transformation, also known as a projective transformation or homography, preserves straight lines (i.e., they remain straight after the transformation).

A 2D perspective transformation is a linear transformation on homogeneous 3-vectors represented by a nonsingular 3 × 3 matrix:

\[
\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad (2.5)
\]

or more briefly, \( x' = Hx \), where \( H \) is a homogeneous matrix \cite{Hartley2003}. There are eight independent ratios among the nine entries of \( H \), and it follows that a projective transformation has eight degrees of freedom.
CHAPTER 2. BACKGROUND

Figure 2.2: A projective rectification of the plane. (a) The given four points on the scene plane, and (b) the rectified plane image. This method does not require knowledge of any of the cameras parameters or the pose of the plane. [Hartley and Zisserman, 2003].

Geometry, in Felix Klein famous “Erlangen Program” [Sharpe, 1997], denotes the study of properties that are invariant under groups of transformations. Shape is deformed under perspective imaging. In the example shown in Figure 2.2(a), the windows are not rectangular in the image, although the originals are. Parallel lines on a scene plane are not parallel any more, but instead they converge to a finite point, which is called “vanishing points”. A central projection image of a plane is related to the original one via a projective transformation, so the image is a projective projection of the original. However this projective transformation can be restored by computing the inverse transformation and applying it to the image. The result will be a new synthesized image where the objects in the plane are shown with their original geometric shape. An example is illustrated in Figure 2.2(b).

2.2 Transformation Estimation Methods

We must stress at this point, that the computation of the transformation matrix $H$ does not require knowledge of any of the camera’s parameters or the pose of the plane. Computation of $H$ can be achieved via point-to-point correspondence techniques, proposed by [Hartley and Zisserman, 2003]. One begins by selecting a quadrilateral region of the image corresponding to a rectangular planar
section of the world. Four point correspondences lead to eight linear equations in the entries of $H$, which are sufficient to solve for $H$ (a total of 8 degrees of freedom). Thus the projective transformation is completely recovered by specifying the position of four points on the plane. The only restriction is that no three points out of the four selected points are collinear. The inverse of the transformation $H$ computed in this way is then applied to the whole image in order to recover the perspective transformation. An example of the aforementioned procedure is shown in Figure 2.2. As proved in [Hartley and Zisserman, 2003], it is not always necessary to know coordinates for four points in order to compute the projective transformation. Some alternative approaches that require less are described in [Hartley and Zisserman, 2003].

Zhang et al., 2010 proposed a method, transform invariant low-rank textures (TILT), for computing the perspective transformation matrix by defining a class of low-rank textures, which capture geometrically meaningful structures in an image. This approach utilizes the cutting-edge convex optimization for recovering of a high-dimensional low-rank matrix and removing corruptions and noise. Many methods have been developed in the past decades to detect and extract invariant features in images, such as invariant points / regions, edges and the widely used scale invariant feature transform (SIFT) Lowe, 2004. These feature detectors more or less are sensitive to local images variations caused by noise, occlusion and illumination. TILT models a class of regular patterns on a planar surface in 3D as low-rank matrices and aims to extract these invariant features as well as the transformation matrix.

While some methods use vanishing points and vanishing lines, some of them use equal length ratios [Hartley and Zisserman, 2003] and orthogonal lines (Figure 2.2). Some superior methods for computing projective transformations are described in [Hartley and Zisserman, 2003] and Liebowitz and Zisserman, 1998. In addition, there is a number of robust rectification algorithms which are based on RANSAC Fischler and Bolles, 1981.

In addition to the transformation in 2D urban images, another challenge is contributed by occlusions and noise. Musialski et al., 2009 proposed a method based on translational and reflective symmetry in façade-images in order to remove occlusions and noise. Musialski et al., 2010a instead uses multi-image stitching to obtain obstacle-free views. An approach proposed by Eisenacher et
al., 2008] aims to generate realistically looking building walls by using example-based texture synthesis.

### 2.3 Façade Analysis and Repetitive Patterns Detection

2D images play an important role in urban scene reconstruction due to the fact that they provide a rich source of information and realism in the final renderings and are easy to acquire, and that there exists an enormous amount of knowledge about their processing. Façade analysis is of particular importance in modeling urban scenes, and has been a very active field of research in the recent decades. In recent years, many interesting approaches for extracting façade texture, repetitive façade structures and façade geometry have been proposed. Some façade-parsing methods aim to automatically subdivide façades into their structural elements. Some of these methods aim at generating an image-based representation of façades, such as panorama imaging and projective texturing. Other interactive façade modeling systems aim at higher quality of details.

One of the fundamental applications of 2D image analysis is panoramas, which are generated for the purpose of visualizing wide landscapes. Some methods generate panoramas by stitching image content from several sources [Burt and Adelson, 1983], [Pérez et al., 2003] [Agarwala et al., 2004], [Agarwala, 2007], [McCann and Pollard, 2008] and [Zomet et al., 2006]. For urban environments, panoramas are usually generated along the path of camera movement [Shum and Szeliski, 2001] [Szeliski, 2006]. Another category of solutions for the generating strip-panoramic images are proposed by [Gupta and Hartley, 1997], [Seitz and Kim, 2003], [Zheng, 2003], [Roman et al., 2004], and [Agarwala et al., 2006].

Another important application of 2D image analysis is for texturing purposes [Musialski et al., 2013]. The problem of generating textures for the interactive rendering of 3D urban models can be addressed by projective texturing from perspective photographs. For instance, [Aliaga et al., 2007], [Sinha et al., 2008] and [Xiao et al., 2008] introduced optimal methods that use projective texture sampling in their modeling pipeline. However, these methods rely on user interaction to guarantee the quality of the results.
CHAPTER 2. BACKGROUND

A series of improvements are further exploited to generate projective texture for existing building models in a fully automatic way, such as the methods proposed in [Coorg and Teller, 1999], [Wang and Hanson, 2001], [Wang et al., 2002], [Böhm, 2004], [Ortin and Remondino, 2005], [Georgiadis et al., 2005], [Grammatikopoulos et al., 2007], [Tan et al., 2008], [Grzeszczuk et al., 2009] and [Kang et al., 2010]. [Korah and Rasmussen, 2007b] presents a method that detects repetitive façade patterns for inpainting façades. A similar interactive approach is proposed by [Pavić et al., 2006] for completing building structures.

Many different approaches have been proposed for façade decomposition that aim at segmenting façades into elements such as doors, windows, and other repetitive patterns. While some methods define façade decomposition as a feature detection challenge, other methods interpret it as an image segmentation problem. Most methods require a pre-processing step for façade image rectification, which is sometimes taken as geometry estimation. On the frontal façades, classic features, such as edge [Canny, 1986], Harris corner [Harris and Stephens, 1988], SIFT [Lowe, 2004] and other features [Bay et al., 2008], are usually detected as basic tools to detect low-level structures.

These low-level features are then employed to infer more sophisticated structures, like floors or windows. However, most earlier attempts were developed on locally splitting heuristics, which is not enough to reliably detect structure in complex façades. Thus many later methods turned to global symmetry structure detection. These approaches often combine low level features with unsupervised clustering, searching and matching algorithms [Hastie et al., 2005]. Other machine learning based methods [Bishop, 2006], [Hastie et al., 2005] aim to match against elements in database and to infer façade structure with predefined shape grammars.

2.3.1 Low-rank Based Façade Analysis

In recent years, matrix factorization methods attracts particular interest for façade image processing. Façade images are usually of low-rank due to many repetitive patterns, such as doors, windows, balconies and tiles. Matrix factorization offers good approximation of low-rank components with a small number of certain basis functions [Musialski et al., 2013]. The approach presented by [Liu et al., 2013a] utilizes factorization for inpainting missing image data. This algorithm is built on stud-
ies about tensor completion using the trace norm and relaxation techniques. Another interesting approach proposed by [Yang et al., 2012b] treats the façade as a matrix and decomposes it into rank one approximations. Before applying the rank one algorithm, the façade must be segmented into multiple rank-one regions by using classification methods.

2.3.2 Heuristic Segmentation

[Wang and Hanson, 2001] and [Wang et al., 2002] proposed a system for the generation of textured models and the detection of windows. They proposed a heuristic oriented region growing algorithm, which iteratively enlarges and synchronizes small seed-boxes until they best fit the windows in the texture. Another use of local image segmentation and heuristics is presented by [Tsai et al., 2005], [Tsai et al., 2006], [Szeliski and Shum, 1997], where a greenness index is calculated in order to detect local mirror axes of façade parts, in order to cover holes left after removing the occluding vegetation. All these methods assume that windows are darker than their surrounding façade.

[Lee and Nevatia, 2004] proposes a segmentation method that utilizes only edges. More specifically, the edges are projected horizontally and vertically to obtain the marginal edge-pixel distributions assuming that window-frames or door boundaries give rise to peaks. A partition of the façade from the thresholded marginal distributions is approximated by a grid. This approach depends very strongly on the parameters of the edge detector, although the partition results are often quite good.

2.3.3 Façade Parsing via Shape Grammars

Façade parsing methods aim at knowledge-based object reconstruction by employing a top-down model that is supposed to be fitted by cues derived from the 2D images. In fact, some methods utilize the concept of inverse procedural modeling. A formal grammar usually is predefined manually or automatically determined from the data in a bottom-up manner. Optimal solutions are described in [Becker, 2009], [Aliaga et al., 2007] and [Ripperda, 2008]. Then the shape grammars are fitted according to the underlying data, which results in very compact representations.

Alegre and Dellaert first proposed the grammar-based segmentation concept in [Alegre and
Dellaert, 2004. They not only introduced a set of rules from a stochastic context-free attribute grammar for modeling the structures, but also proposed a Markov Chain Monte Carlo (MCMC) solution to optimize the parameters. Later on, Mayer and Reznik present a system for façade reconstruction and window detection by fitting an implicit shape model [Leibe et al., 2004], again using MCMC optimization [Mayer and Reznik, 2005] [Mayer and Reznik, 2006] [Mayer and Reznik, 2007].

A series of publications were developed by Brenner and Ripperda in [Brenner and Ripperda, 2006], [Ripperda and Brenner, 2007], [Ripperda, 2008] and [Ripperda and Brenner, 2009], where a system for detecting repetitive façade patterns (especially windows) from images is proposed. In this work, a context-free shape grammar for façades is derived from a set of façade images, and then the Reversible Jump Markov Chain Monte Carlo technique (RJMCMC) is utilized to fit the derived grammar to new models. Becker and Haala presented a system to automatically discover a formal grammar for reconstruction of façades from a combination of LiDAR and image data [Becker and Haala, 2007] [Becker et al., 2008] [Becker, 2009] [Becker and Haala, 2009].

Muller and colleagues [Müller et al., 2007b] proposed a single-view approach for extracting rules from a segmentation of simple regular façades. More specifically, a frontal façade image is split into floors and tiles in a synchronized manner in order to reduce it to a so-called irreducible form, and subsequently fit grammar rules into the detected subdivision. This model is limited to rectilinearly distributed façades and is not able to handle large portion of occlusions. An extension is exploited by [Van Gool et al., 2007] for detecting similarity chains in perspective images and fitting shape grammars in these detected similarity structures.

Pu and Vosselman proposed a higher-order knowledge-driven system for automatically reconstructing façade models from ground laser-scan data [Pu and Vosselman, 2009b]. They extended this work in [Pu and Vosselman, 2009a] and [Pu and Vosselman, 2009c] by combining information from terrestrial laser point clouds and ground images. The system establishes the general structure of the façade using planar features from laser data in combination with strong lines in images.

An automatic approach presented by [Koutsourakis et al., 2009] examines a rectified façade image in order to fit a hierarchical tree grammar. More specifically, the tree formulation of the
façade image is converted into a shape grammar for generating an inverse procedural façade model. This task is formulated as a Markov Random Field in Geman and Geman, 1984. Teboul and colleagues extend this work by combining a bottom-up segmentation through super pixels with top-down consistency checks coming from specific style rules (such as Parisian Style) Teboul et al., 2010. They improved their method by employing reinforced learning in a recent follow-up work Teboul et al., 2011a, Teboul et al., 2013. In their recent work Simon et al., 2012 they present a multi-view approach which build 3D models of buildings by replacing segmented façade elements with pre-defined corresponding 3D structures.

Riemenschneider and colleagues Riemenschneider et al., 2012 proposed an approach which determines the structure of non-trivial façades by using generic grammars and a set of irregular lattices. Martinovic and colleagues Martinović et al., 2012 introduced a method for decomposing the façade into three basic layers of different granularity. This method applies probabilistic optimization to obtain a semantic segmentation of the model.

### 2.3.4 Symmetry Detection

Repeated structures in urban scenes, such as windows, doors and balconies, are sometimes denoted as symmetric structures. Various approaches were introduced to detect these symmetric structure. Early attempts include Reisfeld et al., 1995 and Liu et al., 2004b, where a continuous symmetry transform for images was introduced. Later, Zisserman and colleagues exploited a method for detecting groups of repeated patterns in perspective images. Another similar approach proposed by Turina et al., 2001 aims to detect repetitive patterns on planar surfaces under perspective skew using Hough transform. They also demonstrated that their method works well on building façades. Further work on this topic has been done by Liu and colleagues Liu et al., 2004b. They detected crystallographic groups in repetitive image patterns by using a dominant peak extraction method from the autocorrelation surface. These detected symmetry of regular and near-regular patterns can be utilized by down-stream image processing approaches in order to model new images such as in Hays et al., 2006, Liu et al., 2004c, Park et al., 2011a.

As we all know, most of the building façades are deformed due to an affine or perspective
transformations in 2D urban images. While some methods detect repeated patterns directly on the
transformed images, some methods rectify the image in a pre-processing step.

Some approaches aims at detection of affine symmetry structures in 2D images and 3D point
clouds [Cho et al., 2003, Loy and Eklundh, 2006, Mitra et al., 2006, Podolak et al., 2006]. Similar
methods were then introduced for developing data driven modeling frameworks for symmetrization
and 3D lattice detection in laser-scan of architectures [Mitra et al., 2007, Pauly et al., 2008,
Mitra et al., 2010]. Other methods detect symmetric structures directly in perspective images.
For instance, the method proposed by [Wu et al., 2010b] focuses on detecting grid-like symmetry
in façade images under perspective skew. The detection results were then utilized to reconstruct
dense 3D structure in a follow-up work [Wu et al., 2011]. Later, Park and colleagues [Park et al.,
2011b] introduced a method to detect translational symmetry for determining façades.

In recent years, a set of approaches were proposed for detecting symmetry in 3D ground-based
urban laser-scan data [Berner et al., 2008, Berner et al., 2011, Bokeloh et al., 2009]. Shen and
colleagues [Shen et al., 2011] proposed a heuristic segmentation method for detection of symmetry
and repetitions. This method segments LiDAR scans of façades and detects concatenated grids, and
automatically partitions the façade in an adaptive manner to generate a hierarchical representation
of the input architecture.

Another category of methods tackles the detection of repeated structures on frontal façade
images. Korah and Rasmussen introduced a method in [Korah and Rasmussen, 2007a] for automatic
detection of 2D grids on façade images. Other methods, such as [Wenzel et al., 2008, Musialski
et al., 2010b], detect rectilinear patterns in rectified façade images using local features. Zhao and
Quan introduced a similar method to detect 2D symmetric lattice structure in [Zhao and Quan,
2011]. Later on, they extended this method to detect symmetries in [Zhao et al., 2012].

Alsisan and Mitra [Alsisan and Mitra, 2012] propose an approach that combines grid-detection
and a MRF-regularization in order to provide variation-factored façade representation. A similar
approach proposed by Tylecek and Sara [Tyleček and Šára, 2011] aims to detect grids of windows
in rectified façade images using a MCMC optimization method. Other recent frameworks for
detection of regularly distributed façade elements and segmentation of LiDAR façade data include
Recently, a set of publications have been developed for detecting symmetry across multiple registered urban images \cite{Jiang2011,Ceylan2012}, where the results can be utilized to recover missing structure of buildings.

### 2.3.5 Machine Learning Methods

Another group of methods aims at detection of windows and other pre-specified structural elements using machine learning methods. For example, Schindler and Bauer \cite{Schindler2003} proposed a method that utilizes supervised learning in order to match shape templates against dense point clouds. The method proposed by Mayer and Reznik \cite{Mayer2007} models the façades by matching template images from a manually constructed window image database. A set of approaches presented in \cite{Ali2007,Drauschke2008, Schapire1999} combine template matching with machine learning by training a classifier and are able to identify most of windows in perspective images.

Cech and Sara \cite{Cech2008} proposed an approach that detects strictly axis-aligned rectangular pixel configurations in a MRF model. Haugeard and colleagues \cite{Haugeard2009} exploited a method that extracts rectangular windows in the façade image and utilizes the detection results for retrieving similar windows in a database of similar images of façades. Features learning has also been used for the same purpose, such as the user-supervised technique presented by \cite{Sunkel2011} that learns line features in geometric models. Other façade segmentation methods include \cite{Zhao2010} and \cite{Dai2012}.

### 2.3.6 Interactive Façade Modeling Methods

In the previous sections, we present an overview of automatic façade analysis approaches. While automatic methods seem to fast and scalable, they share the same disadvantages that the output models are lack of high quality and level of detail. Interactive approaches, on the other hand, can tackle this problem by generating better quality and higher level of detail. Xiao and colleagues \cite{Xiao2008} introduced an interactive image-based approach for façade modeling that utilizes images captured along streets. This method relies on structure from motion as a source for camera
parameters and requires a considerable amount of user interaction in order to correct misinterpretations of the automatic routines. Hohmann and colleagues [Hohmann et al., 2009] proposed a generative modeling language (GML) shape grammar based system for modeling the façade structures. The concept of GML shape grammar is introduced in [Havemann and Fellner, 2005]. Another similar work was proposed by [Aliaga et al., 2007], where grammar rules are determined manually against the façade images and can be used for procedural modeling of similar buildings. Nan and colleagues [Nan et al., 2010] proposed an interactive method that reconstructs façades from terrestrial LiDAR data based on semi-automatic snapping of small structural assemblies. Another interesting semi-automatic system described in [Musialski et al., 2012] detects significant elements in orthographic façade images using unsupervised clustering and is able to model high quality and level of detail in competitive time.

2.4 3D Reconstruction

3D scene reconstruction is a task of generating 3D models of a scene given multiple 2D photographs taken of the same scene, and has been one of the longest studied problems in computer vision. The problem is challenging due to the limitations given during the data acquisition process. It is often difficult to acquire coherent and complete data of urban environments, as buildings are often located in narrow streets surrounded by other buildings and obstructions. Thus it is impossible to acquire complete 2D photographs, or 3D scans either from the ground or from the air. In addition, unwanted objects in front of the buildings, such as trees, street signs, vehicles and pedestrians, can cause a large number of occlusions on 2D images and “holes” on the acquired 3D data. Thus, recovery of the original 3D structure that has been corrupted through such obstructions is particularly challenging.

We believe that the façade structure analysis results provide important structural information for 3D reconstruction and many 2D façade analysis methods, such as low-rank based methods and shape grammars, can be applied to 3D point clouds. For example, Vanegas and colleagues [Vanegas et al., 2010] proposed grammar-driven methods for automatic building generation from air-borne imagery. An iterative approach proposed by [Aliaga et al., 2007] aims to visualize a realistic and
textured urban model based on a repertoire of grammars extracted from a set of photographs of a building. A combination of procedural modeling with GML and shape grammars [Havemann and Fellner, 2005] is also adapted in a 3D modeling system proposed by [Hohmann et al., 2009] and [Hohmann et al., 2010].

Stamos and Allen [Stamos and Allen, 2000a], [Stamos and Allen, 2000b], [Stamos and Allen, 2001], [Stamos and Allen, 2002] developed a series of publications where a system was proposed for reconstruction of buildings from sets of range scans combined with sets of unordered photographs. Their method is based on fitting planar polygons and other features into pre-clustered point clouds. Bauer and colleagues [Bauer et al., 2003] proposed a similar approach for the detection and partition of planar structures in dense 3D point clouds of façades. A number of image-based modeling methods were recently proposed for exploration and reconstruction of urban environments. For example, Snavely and colleagues [Snavely et al., 2006], [Snavely et al., 2008b], [Snavely et al., 2008a], [Snavely et al., 2010] developed a system “Photo Tourism” that enables navigation through large collections of registered photographs. In all the reconstruction methods mentioned above, detection of façade elements provides rich structure information, such as the locations and frames of windows and doors.
Chapter 3

Rectifying 2D Façade Images

In this chapter, we describe algorithms that aim to rectify façade images and extract low-rank textures. The next step, detecting repeated patterns, is based on the rectification results which are presented in this chapter. Facade image rectification essentially is a problem of finding the orientation of building facades. Image rectification plays an important role in lots of recent computer vision research, such as 3D reconstruction [Criminisi et al., 2000] [Yang et al., 2005], symmetry or repeated patterns detection [Zhao and Quan, 2011], [Zhao and Quan, 2011] and [Teboul et al., 2011b], scene understanding and facade parsing. Many of recent research works achieved success in detection of repeated patterns in 2D frontal facade images. Rectifying facade images is an important preprocessing step in this methods. In this chapter, our goal is to automatically rectify the facade images in an efficient and effective way. The task is challenging due to the fact that the acquired 2D images are generally occluded by noise, like trees, traffic noise, pedestrians and illumination shadows. The problem becomes challenging when the amount of noise and occlusion increases (see the example in Figure 3.1).

Parallel lines in real urban scenes are the most prominent features and converge to vanishing points in 2D images. Thus vanishing points are largely utilized in alignments. Early research success has been achieved by exploiting vanishing points [Criminisi et al., 2000], Canny edges [Canny, 1986], SIFT [Lowe, 1999] and Harris corners [Harris and Stephens, 1988].

Another category of methods use intrinsic and extrinsic parameters of the 2D cameras which
have taken those images. When the parameters are not available, this can be achieved by estimating the camera pose, such as focal length, by using features.

In recent years, the robust Principle Component Analysis based methods attracted a lot of attention. These methods use the raw pixel information of a single image and compute the transformation matrix accurately. \cite{Zhang et al., 2010} proposed an algorithm, Transform Invariant Low-rank Textures (TILT), which takes a user-defined bounding box for low-rank texture as input, and outputs a rectified object by computing the perspective transformation matrix based on a manually selected input bounding box.

However, big façades (such as the skyscrapers in New York City) can be largely distorted in 2D images when being taken by cameras. The distortion affects the accuracy of vanishing points detection, thus a purely feature-based method usually is not able to rectify the façade completely. In addition, TILT relies on user-defined façade textures as input, which largely draws back the performance and automatic mechanism. Another drawback is that TILT utilizes a simple branch-and-bound strategy to initialize the transformation matrix. This initialization strategy can be largely improved in urban images, since images of urban scene contain rich structural information, such as parallel lines from windows, doors and building boundaries.

In this chapter, we propose a method to rectify the image precisely and automatically by combining the traditional feature-based methods and robust Principle Component Analysis (PCA) \cite{Candes et al., 2011}. This approach takes advantages of the information offered by both vanishing points and robust PCA, but avoiding their weaknesses. We also demonstrate the effectiveness and efficiency by running experiments.

### 3.1 Algorithm Overview

Our strategy is to automatically detect a façade region in a 2D urban image and rectify the façade. We represent a façade region as a low-rank texture, as most of the building façades are designed in low-rank structure in real world. As shown in Figure 3.1, the real façade texture is of low-rank in Figure 3.1(c), and it is of full rank in both Figure 3.1(a) and (b) due to the occlusion and
CHAPTER 3. RECTIFYING 2D FAÇADE IMAGES

Figure 3.1: An image rectification example. (a) The original image, on which the red box is the input of our algorithm and the green quadrilateral indicates the transformation. (b) The rectified image texture. (c) The low-rank component detected by our algorithm, where \( \text{rank} = 6 \).

transformation. Given a 2D urban image where building façades are included, we first find a low-rank façade region by exploiting traditional features: Harris corners [Harris and Stephens, 1988] and vanishing points [Horn, 1986]. Then we develop a robust PCA algorithm to rectify the façade completely.

The input of this step is a 2D image of a building façade. The output is a transform matrix as well as a representative texture on the façade. This representative texture is essential for an automated system, since it is used as input by TILT [Zhang et al., 2010] (this automation provides a performance improvement of 19.6\% over manual selection; see the comparison in Table 3.2).

First of all, a vanishing point detection (VPD) algorithm is employed, and the major vanishing points are obtained (two are required). Based on the vanishing points detected, a block division algorithm follows. A block is defined as a quadrilateral formed by the vanishing point directions. We then detect the Harris corners for the input façade image by applying a Harris Corners detection method. By counting the number of Harris corners within each block and calculating the variances of the neighboring blocks our algorithm is able to select a representative texture. By representative here we mean a texture that contains a significant periodic structure. Finally, the detected vanishing points are used for the generation of a transform matrix for image rectification.

Next, we exploit algorithms that aim to get the rectified building façade based on low-rank
techniques by combining vanishing points. Vanishing points provide rich information about the orientation of building façades in urban scene, and there is a simple and efficient algorithm based on vanishing points to detect the orientation of building façades. However, most of the big buildings in 2D images are distorted when being acquired by 2D cameras. Thus the vanishing points may shift somehow when the images are distorted. This deviation directly causes an inaccurate rectification. Thus we develop a robust PCA based algorithm as a complementary algorithm. Experiments show that the combination of those two methods largely improves the performance.

3.2 Detecting Features

3.2.1 Vanishing Points

Vanishing points provide rich information about the orientation of building façades and thus can rectify façades when the façades are not distorted. However, distortion is particularly associated with camera lens. Skyscrapers, captured by high field-of-view lens, are usually largely distorted. Purely vanishing points based methods are affected by the distortion and are not able to rectify the façades precisely. Although vanishing points sometimes fail to rectify the façades precisely, they can provide a rough approximation of the transformation matrix that rectifies the image. Thus we use vanishing points to initialize the robust PCA algorithm. Besides the initialization, vanishing points serve another purpose for detecting low-rank textures, by integration with other features (such as Harris corners), as will be shown in the next sections.

3.2.2 Harris Corners

Corner detection is frequently used to extract certain features and infer the contents of an image within computer vision systems, such as in 3D modeling, object recognition, motion tracking and image registration. A corner is defined to be an interest point with low self-similarity \cite{Moravec1980}. A corner can be an isolated point of local intensity maximum or minimum, line endings. We observe that Harris corners are distributed almost uniformly in non-occluded façade areas that contain repeated patterns. Otherwise, occlusions like trees, pedestrians and traffic lights, may
produce a non-uniform distribution of the Harris corners. For example, while the Harris corners in a tree area are very dense due to the non-uniform branches, the Harris corners in sky areas can be very sparse due to the lack of local color intensity difference. Thus the distribution of Harris corners provides information about the building façades localization. A corner detection approach frequently used is first proposed by Harris and Stephens \cite{Harris and Stephens, 1988}, which in turn is an improvement of a method by \cite{Moravec, 1980}.

3.3 Automatically Selecting A Façade Region

The inputs of this step include a 2D image of a building façade and the features detected in last section. The output is a representative texture on the façade as well as a transform matrix that is used to initialize the façade rectification. This representative texture is essential for an automated system, since it is used as input of both our urban low-rank algorithm and the low-rank algorithm (named TILT) in \cite{Zhang et al., 2010} (this automation provides a performance improvement of 19.6% over manual selection; see the comparison in Table 3.2). The algorithm is implemented in three steps: (1) feature extraction, (2) block division, and (3) transformation initialization and representative texture selection.

First of all, we extract Harris corners \cite{Harris and Stephens, 1988}. We also detect the two major vanishing points by using the method of \cite{Li et al., 2010}. We then divide the façade into blocks (quadrilateral) along vanishing points directions. Finally, we compute the homography matrix that rectifies the image and select the representative texture by combining Harris corners distribution information within the detected blocks. We observed that Harris corners are distributed almost uniformly in unoccluded façade areas that contain repeated patterns. Otherwise occlusions may produce a non-uniform distribution of the Harris corners. For example, the Harris corners in a tree area will be very dense and non-uniformly spaced.

In particular, starting from each detected vanishing point we draw hypothetical lines at angular intervals towards the image assuring that all Harris corners are included in the generated quadrilaterals (see Figure 3.2(b)). This is achieved by computing the smallest angles $\theta_1$ and $\theta_2$ (one for
each vanishing point) that ensure inclusion of all Harris corners. Then, we divide each range $\theta_i$ into $m$ parts. The intersections of the imaginary lines thus create $m \times m$ quadrilaterals.

We observed that the number of corners in each block does not change much after perspective distortion of the façade image, although the distribution of the Harris corners depends on the location of the vanishing points. Excluding strong perspective distortions is not so crucial (in such cases even robust techniques fail to rectify the image). We thus assume that the ideal texture should consist of neighboring blocks that have a similar and uniform distribution of Harris corners. We then count the number of Harris corners in each block, and get an $m \times m$ matrix $C$, where each element $C_{i, j}, i, j = 1, \cdots, m$ is the number of Harris corners in the corresponding block.

In order to isolate the $r \times c$ submatrix of $C$ containing the most representative texture of the given façade, let us consider that its elements are random samples from a double exponential distribution. We then compute the sample median $\mu_C$ of the elements of matrix $C$. Finally, we slide a window of size $r \times c$ along matrix $C$, and compute in each location the sample mean deviation from the sample median, that is:

$$ S_{i,j} = \frac{1}{rc} \sum_{k=i}^{i+r-1} \sum_{l=j}^{j+c-1} |C_{k, l} - \mu_C| $$

thus forming a score matrix of size $(m - r + 1) \times (m - c + 1)$. In all of our experiments, we set $m$ to 10 and fix $r$ and $c$ to a given percentage of $m$, that is $r = c = 0.4m$.

It is well known that the sample median and the sample mean deviation from the sample median are the maximum likelihood estimators of the mean and standard deviation of the distribution. Thus, by choosing the sliding window with the highest score we actually choose the one with the minimum variance among all the best likelihood estimators of the mean value. This window will be selected and used as the input of the TILT algorithm in order to get the low-rank component and rectification of the façade image (see Figure 3.2(h) for an example).
Figure 3.2: Texture selection procedure. (a) The blue lines denote the directions of the major vanishing points. (b) Hypothetical lines drawn from vanishing points towards image plane. (c) 10 by 10 blocks divided along vanishing points directions. (d) Harris corners features extracted for block selection. (e) The yellow box localizes a raw texture selected by the algorithm. (f) The green quadrilateral is the isolated yellow area in (e). (g) A rectified façade image through a pure vanishing points algorithm. (h) The largest rectangular box that fits inside the selected region, to be used for initialization. (i) A rectified façade by the urban low-rank algorithm with the red box in figure (e) as input texture.
3.4 The Urban Low-rank Algorithm for Façade Image Rectification

Low-rank matrix recovery and approximation algorithms have been extensively stated lately for their importance in theory and practice. Such matrices arise in many real data analysis problems when the high dimensional data of interest lies on a low dimensional linear subspace. Principal Component Analysis (PCA) is one of the most popular algorithms to compute low-rank approximations of a high-dimensional data matrix. Basically, PCA solves the following optimization problem:

$$\min_{X \in \mathbb{R}^{N \times M}: \text{rank}(X) \leq k} \|A - X\|_2^2.$$  \hspace{1cm} (3.2)

Specifically, if $k \leq r = \text{rank}(A)$ and we define the matrix

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i$$ \hspace{1cm} (3.3)

where $\sigma_i$, $u_i$ and $v_i$ denote the $i$-th singular value, right and left singular vector respectively then, we have that

$$\|A - A_k\|_2^2 = \min_{X \in \mathbb{R}^{N \times M}: \text{rank}(X) \leq k} \|A - X\|_2^2.$$ \hspace{1cm} (3.4)

That is, matrix $A_k$ minimizes the $l_2$ norm existing between matrix $A$ and any rank $k$ approximation of this matrix.

However, although the truncated SVD is widely used, singular vectors $u_i$ and $v_i$ may lack any meaning in terms of the properties of the data. For example, it is well known that singular vectors play a very important role in the Karhunen Loeve (KL) transform when the data are drawn from a Gaussian distribution. Specifically, it is well known that matrix $A_k$ defined in Eq. (3.3) offers the optimal solution when matrix $A$ is corrupted by i.i.d Gaussian noise. The major drawback of PCA is its sensibility to errors of large magnitude even if matrix $A$ is contaminated with such errors in a very small part. In fact, a single corrupted entry can throw the low-rank matrix $A_k$ arbitrarily far from the true solution.
CHAPTER 3. RECTIFYING 2D FAÇADE IMAGES

3.4.1 Problem Formulation

PCA has been one of the most widely used statistical tool for data analysis and dimensionality reduction today. The problem modeling of low-rank texture recovery was first proposed in [Zhang et al., 2010]. The basic idea is to view each 2D image as a matrix and seek a transformation that gives rise to a low-rank matrix subject to sparse errors.

**Low-rank texture** A $m \times n$ image $I^0$ of a texture is considered to be of low-rank if the rank of the matrix $I^0$ is much smaller than both $m$ and $n$. We observe that a very rich class of regular patterns exist on a planar surface in 3D urban scene, such as windows, doors, balconies and tiles, which can be modeled approximately as a low-rank matrix (see Figure 3.3 for some examples).

**Deformed low-rank texture** Although many structures in 3D scene exhibit low-rank textures, their appearance is transformed in the captured 2D images due to the viewpoints of the 2D cameras. Suppose $I^0$ is a low-rank texture that lies on a planar surface in the scene, the image $I$ we observe from a certain viewpoint is a transformed version of $I^0$:

$$I = I^0 \circ \tau^{-1} = I^0(\tau^{-1}),$$

where $\tau$ represents the transformation matrix. In this chapter, we assume $\tau$ is either a rotation matrix, an affine matrix or a homography. In general, the transformed texture $I$ is no longer a low-rank matrix. For example, a horizontal edge has rank one, but it becomes a full-rank
Corrupted Low-rank Texture In addition to domain transformations, the 2D images of low-rank textures might be corrupted by noise and occlusion or contain some pixels from the surrounding background. Such deviations can be modeled by an error matrix $E$:

$$ I = I^0 + E. $$

As a result, the image $I$ is not of low rank. To make the problem meaningful, we assume that the low-rank component $I^0$ is not sparse. Since if $I^0$ is both sparse and low-rank, we can not decide whether it is low-rank or sparse. Another issue arises if the sparse matrix has low-rank. This will occur if, say, all the nonzero entries of $E$ occur in a column or in a few columns. Then it is clear that we are not able to recover $I^0$ and $E$ by any method, since $I = I^0 + E$ would have a column space equal to $I^0$. In order to avoid such meaningless situation, we assume that only a small fraction of the image pixels are corrupted, and hence, $E$ is a sparse matrix and the sparsity pattern of the error matrix $E$ is selected uniformly at random. A more detailed explanation can be found in [Hubert and Engelen, 2004].

Our goal is to simultaneously recover the domain transformation matrix $\tau$ and the low-rank texture $I^0$ from an deformed and corrupted image $I = (I^0 + E) \circ \tau^{-1}$. The problem is precisely
CHAPTER 3. RECTIFYING 2D FAÇADE IMAGES

modeled in [Zhang et al., 2010] as follows:

**Problem 1** Given a deformed and corrupted image of a low-rank texture: \( I = (I^0 + E) \circ \tau^{-1} \), recover the domain transformation \( \tau \).

The above problem formulation leads to the following optimization problem:

\[
\min_{I^0, E, \tau} \text{rank}(I^0) + \lambda \|E\|_0 \quad \text{s.t.} \quad I \circ \tau = I^0 + E, \tag{3.5}
\]

where \( \|E\|_0 \) is the \( L_0 \) norm which denotes the number of non-zero entries in \( E \), and \( \lambda \leq 0 \) is a weighting parameter that trades off the rank of the texture versus the sparsity of the error. That is, we aim to recover the \( I^0 \) of the lowest possible rank and the error matrix \( E \) with the fewest possible nonzero entries that agrees with \( I \) up to a transformation matrix \( \tau \).

### 3.4.2 Solve the Problem via Alternating Direction Methods

As proposed in [Peng et al., 2012] and [Zhang et al., 2010], the rank function and \( L_0 \) norm are generally NP-hard and thus they are extremely difficult to optimize. However, recent research breakthroughs have shown that the rank function and \( L_0 \) norm can be replaced by their convex surrogates, the matrix nuclear norm and the \( L_1 \) norm respectively, which leads to the following optimization problem:

\[
\min_{I^0, E, \tau} \|I^0\|_* + \lambda \|E\|_1 \quad \text{s.t.} \quad I \circ \tau = I^0 + E. \tag{3.6}
\]

This optimization problem defined above is not convex as the constraint \( I \circ \tau = I^0 + E \) is nonlinear, although the objective function \( \|I^0\|_* + \lambda \|E\|_1 \) is convex.

Thus we can linearize the constraint function as follows [Zhang et al., 2010]:

\[
I \circ \tau + \nabla I \Delta \tau = I^0 + E, \tag{3.7}
\]

which leads to the following optimization problem:
\[
\min_{I^0, E, \tau} \left\| I^0 \right\|_* + \lambda \left\| E \right\|_1 \quad \text{s.t.} \quad I \circ \tau + \nabla I \triangle \tau = I^0 + E. \tag{3.8}
\]

However, there is no need to compute the three parameters \( I^0, E \) and \( \tau \) simultaneously. We observe that by fixing the transformation \( \tau \), we can iteratively find the best estimation of \( I^0 \) and \( E \) and then re-compute the new transformation \( \tau \) based on the updated \( I^0 \) and \( E \). Instead of linearizing the constraint, we use a normalized constraint for solving \( I^0 \) and \( E \) efficiently:

\[
\frac{I \circ \tau}{\left\| I \circ \tau \right\|_F} = \frac{I^0 + E}{\left\| I^0 + E \right\|_F}. \tag{3.9}
\]

In order to solve the convex optimization problem (3.8), we propose an iterative strategy based on the augmented Lagrangian. This iterative algorithm updates \( I^0 \) and \( E \) simultaneously for each \( \tau_{k-1} \). Then \( \tau_k \) is updated separately using the updated \( I^0_k \) and \( E_k \).

The efficiency and accuracy has been demonstrated by sufficient experiments. We define the augmented Lagrangian for solving \( I^0 \) and \( E \) as:

\[
L_\mu(I^0, E, Y) = f(I^0, E) + \langle Y, R(I^0, E, \Delta \tau) \rangle + \frac{\mu}{2} \left\| R(I^0, E, \Delta \tau) \right\|_F^2. \tag{3.10}
\]

where \( \mu > 0 \), \( Y \) is a Lagrange multiplier matrix, \( \langle \cdot, \cdot \rangle \) denotes the matrix inner product, \( f(I^0, E) = \left\| I^0 \right\|_* + \lambda \left\| E \right\|_1 \), and \( R(I^0, E, \Delta \tau) = \frac{I \circ \tau}{\left\| I \circ \tau \right\|_F} - \frac{I^0 + E}{\left\| I^0 + E \right\|_F} \). Thus, a basic iteration scheme for the problem defined in (3.8) is given by

\[
(I^0_k, E_k) = \arg\min_{I^0, E} L_{\mu_k}(I^0, E, Y_{k-1}), \quad Y = Y_{k-1} + \mu_{k-1} R(I^0, E_k, \Delta \tau). \tag{3.11}
\]

So far, we only solve for the variables \( I^0 \) and \( E \), and leave \( \tau \) to be updated later. By taking \( \tau \) as a fixed value, the constraint in Eq. (3.8) is already linear, and there is no need to linearize it by adopting a new term \( \nabla I \triangle \tau \) while solving \( I^0 \) and \( E \). And based on the above strategy, we conclude the solution, the urban low-rank algorithm, in Algorithm I.
Algorithm 1: Urban Low-Rank Algorithm.

Input: a image texture $I$, a weight factor $\lambda$ and an initial homography $\tau = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

1: repeat
2: Normalization and compute Jacobian:
   $I \circ \tau \leftarrow \frac{I \circ \tau}{\|I \circ \tau\|_F}$;
   $I^0 + E \leftarrow \frac{I^0 + E}{\|I^0 + E\|_F}$;
   $\nabla I \leftarrow \frac{\partial}{\partial \tau} \left( \frac{\text{vec}(I \circ \tau)}{\|\text{vec}(I \circ \tau)\|_F} \right)_{\tau=\tau_0}$;
   $\Delta \tau = 0$.
3: Solve the linear problem:
   $\min_{I^0, E} \|I^0\|_* + \lambda \|E\|_1$, s.t. $\frac{I \circ \tau}{\|I \circ \tau\|_F} = \frac{I^0 + E}{\|I^0 + E\|_F}$.
4: Initialization: $k = 0, Y_0 = 0, E_0 = 0, \mu_0 > 0, \rho > 1$
5: Inner loop: iteratively approximate the optimal solution for $I^0$ and $E$ by using Algorithm 2
6: Compute $\Delta \tau$ and update $\tau$:
   $\Delta \tau = (\nabla I)^T (I + E - I \circ \tau)$;
   $\tau \leftarrow \tau + \Delta \tau$;
7: until convergence
8: Output: $I^0, E, \tau$.

Algorithm 2: Inner loop for updating $I^0$ and $E$. Input: $k = 0, Y_0 = 0, E_0 = 0, \mu_0 > 0, \rho > 1, \tau$

1: repeat
2: $(U_k, \Sigma_k, V_k) = \text{svd}(I \circ \tau - E_k + \mu_k^{-1}Y_k)$;
   $I^0_{k+1} = U_k S_k \Sigma_k^{-1} V_k^T$;
   $E_{k+1} = S^{-1}_k \mu_k^{-1} [I \circ \tau - I^0_{k+1} + \mu_k^{-1}Y_k]$;
   $Y_{k+1} = Y_k + \mu_k (I \circ \tau - I^0_{k+1} - E_{k+1})$;
   $\mu_{k+1} = \rho \mu_k$;
3: until convergence
4: Output: $I^0_{k+1}, E_{k+1}$.

3.5 Experiments and Analysis

In order to evaluate the performance, we first compare our urban low-rank algorithm and TILT by using exactly the same manually selected input texture and branch-and-bound initialization transformation matrix and run them on a computer with an 1.8 GHz Intel Core i7 CPU and a
Figure 3.5: Comparison between TILT and our urban low-rank algorithm. (a) The red box is the input texture, and the green box denotes the output of our algorithm, (b) the rectified facade by our urban low-rank algorithm, (c) the red box is the input for TILT, and green quadrilateral denotes the transformation $\tau$ recovered by TILT, and (d) the final rectification result from TILT, where TILT failed to rectify this image.
CHAPTER 3. RECTIFYING 2D FAÇADE IMAGES

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Rank</th>
<th>Run Time</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>4</td>
<td>58.6783s</td>
<td>72.0%</td>
</tr>
<tr>
<td>TILT</td>
<td>40</td>
<td>127.6698s</td>
<td>63.5%</td>
</tr>
</tbody>
</table>

Table 3.1: Speed Comparison of TILT and our urban low-rank algorithm, with branch-and-bound initialization and manually selected texture.

<table>
<thead>
<tr>
<th>Initialization method</th>
<th>Run Time</th>
<th>Success Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch-and-bound</td>
<td>36.63s</td>
<td>65%</td>
</tr>
<tr>
<td>Vanishing points based</td>
<td>38.6s</td>
<td>84.6%</td>
</tr>
</tbody>
</table>

Table 3.2: Performance comparison between the branch-and-bound initialization and vanishing points based transform initialization. The vanishing points initialization results in significant increase in success rate without a penalty in speed. The experiments are done on a set of 306 façade images.

4GB 1333 MHz DDR 3 memory. Then we run both of these two algorithms on the same computer with automatically selected input textures with vanishing points-based transformation initialization. Tables 3.1 and 3.2 show the results for the two comparison experiments, respectively. In total, we tested 182 urban images shot in New York City and 100 sample images provided by TILT database, in different architecture styles, and containing different percentage of occlusion pixels. Out of the 182 urban buildings in NYC, TILT gets 114 rectified correctly and ours gets 128 successfully rectified. Out of the 100 images in TILT database, our algorithm successfully finds the correct transformation information on 75 ones, while TILT gets 65. A typical failure case is shown in Figure 3.5. All experiments were conducted during 2012.

As shown in Table 3.1, our algorithm succeeds in 72.0% out of the total 282 images, while TILT succeeds only in 63.5%. A separate experiment demonstrates that by using the vanishing points as initialization, the accuracy can be improved by up to 19.6%. This experiment is run on 306 images in total. The results in Table 3.2 clearly state that in urban environments, the use of vanishing points significantly improves the quality of results. The urban images we use for test include 182 façade images we collected in New York City as well as 124 sample façade images from TILT’s web resources.

The low-rank components recovered by our algorithm have closer rank to the actual structure in the images. As shown in Table 3.2, the average rank of the low-rank components from our
algorithm is 4, while the average rank of the low-rank components from TILT is 40. The ideal rank of input textures ranges from 2 to 5, thus a low-rank component with higher rank contains more noise. This improvement is contributed by the automatic error rate $\lambda$ controlling strategy in our method. TILT assumes that all input image textures have an error rate $\frac{1}{\sqrt{\min\{m,n\}}}$, where $m$ and $n$ respectively denote the number of rows and columns of the input texture.

**Range of Convergence** In order to evaluate the range of convergence of our algorithm, we run our algorithm on a series of transformed standard checkerboard pattern. The results are shown in Figure 3.6 and 3.7.
Figure 3.7: Range of convergence for affine transformation. \( x \)-axis: rotation angle \( \theta \). \( y \)-axis: skew parameter \( t \). The entire region on and below the line indicates success in all trials while the region above the line indicates failure in all trials.
Chapter 4

Kronecker Product Model: Theory

4.1 Introduction

In this chapter, we will introduce a novel theoretical model, which integrates low-rank textures, robust PCA and the Kronecker Product, to model a frontal façade structure globally in an efficient and effective way. The input of this method is a frontal façade texture, which actually is the output of the approach described in Chapter 3. The output not only recovers the repeated patterns, but also removes noise and occlusions. We separately state the implementation of this algorithm in the next chapter.

4.1.1 Motivation

In recent years, the task of façade parsing attracted a lot of interest in the context of urban scene reconstruction. While some of these methods aim at detecting the symmetric structure directly on the perspective façade images, other methods interpret the repeated pattern detection problem on rectified frontal-facades. Most of these methods first employ conventional local invariant features, like corners, edges, Difference of Gaussian (DoG) as well as SIFT. They then try to find the repeated or symmetric structures by looking for the pointwise correspondence of the detected features.

The façade parsing problem becomes especially challenging when the repeated patterns are heavily occluded by noise. In this situation, traditional local features based methods may fail to
find the desired patterns underneath the occlusions. Thus numerous invariant features and descriptors have been proposed, studied and compared in the literature. A very interesting invariant feature, low-rank texture, was recently proposed by [Zhang et al., 2010]. The low-rank textures capture geometrically meaningful structures in an image, and a more important advantage is that they encompass conventional local features such as edges and corners as well as all kinds of regular, symmetric patterns ubiquitous in urban environments. In Chapter 3 we introduced a low-rank based method that aims at rectifying a perspective façade image and finding the domain transformation. Besides this application, we believe that it potentially is a very powerful tool that will allow people to accurately extract rich structural and geometric information about the 3D scene from its 2D images. It has been proved that low-rank textures are truly invariant of image domain transformations. Yang adopted this low-rank feature to detect repeated patterns on frontal façade images [Yang et al., 2012a]. However, that work still has to first perform classifications and segment the façade into multiple rank-one patches. The output largely relies on the classification step, and thus does not fully take the advantage of the low rank property of the urban structures.

4.1.2 The Kronecker Product

In mathematics, the Kronecker product, denoted by $\otimes$, is an operation on two matrices of arbitrary size resulting in a block matrix. If $A$ is an $m \times n$ matrix and $B$ is a $p \times q$ matrix, then the Kronecker product of $A$ and $B$ is the $mp \times nq$ matrix

$$A \otimes B = \begin{bmatrix}
    a_{11}B & a_{12}B & \cdots & a_{1n}B \\
    a_{21}B & a_{22}B & \cdots & a_{2n}B \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{m1}B & a_{m2}B & \cdots & a_{mn}B
\end{bmatrix}$$

It is a generalization of the outer product, which is denoted by the same symbol, from vectors to matrices, and gives the matrix of the tensor product with respect to a standard choice of basis. The Kronecker product is named after Leopold Kronecker. In the past, the Kronecker product was sometimes called Zehfuss matrix, after Johann George Zehfuss who introduced the matrix operation
CHAPTER 4. KRONECKER PRODUCT MODEL: THEORY

in 1858.

Although Kronecker product is widely used and well studied in mathematics, signal processing, computer vision networks [Leskovec et al., 2010] [Mipiperis et al., 2008] [Van Loan, 2000], no one has introduced it to façade modeling before. The original idea of modeling façades is motivated by the elegant structure and rich algebraic properties of Kronecker product.

4.2 Modeling a Building Façade via Kronecker Products

In this section we describe our Kronecker product modeling approach that is applied on a rectified façade image. It is a novel representation that describes a large subset of façade examples.

4.2.1 Ideal Façade Modeling

To this end, let us consider the partition of all ones orthogonal array \(1_{l_v \times l_h}\) of size \(l_v \times l_h\) by using the following mutually exclusive, \(1 - 0\) matrices \(M_k\), \(k = 1, 2, \cdots, K\) of size \(l_v \times l_h\) each, that is:

\[
< \text{vec}\{M_k\}, \text{vec}\{M_l\} > = \begin{cases} 
||\text{vec}\{M_k\}||_0, & k = l \\
0, & k \neq l
\end{cases} \tag{4.1}
\]

\[
\sum_{k=1}^{K} M_k = 1_{l_v \times l_h} \tag{4.2}
\]

where \(\text{vec}\{X\}\), \(< x, y >\) and \(||x||_0\) denote the column-wise vectorization of matrix \(X\), the inner product of vectors \(x, y\) and the \(l_0\) norm of vector \(x\) respectively. As it is clear from Eqs. (4.1,4.2), different choices of matrices \(M_k\) result in different partitions of orthogonal block \(1_{l_v \times l_h}\). Let us now associate with each component \(M_k\), \(k = 1, 2, \cdots, K\) of the partition of array \(1_{l_v \times l_h}\) defined in Eq. (4.2), a 2-D pattern \(P_k\) of size \(N_v \times N_h\) that is going to be repeated according to \(M_k\). The patterns should have a piecewise constant surface form. In particular, with the aim of patterns \(P_k\) several windows, doors and/or balconies of different architectures can be formed.

\[^1\text{Matrices that contain only combinations of 1s and 0s}\]
We can now define a subset of urban building façades that can be expressed as a sum of Kronecker products:

\[
\mathcal{F}_{N \times M} = \sum_{k=1}^{K} \lambda_k (M_k \otimes P_k) \tag{4.3}
\]

where \( \lambda_k, k_1, 2, \cdots, K \) are weights that represent the brightness for each group of patterns.

Finally, suppose that \( N \times M \) is the size of the urban building façade image. By the definition of the Kronecker product it is obvious that \( N = l_v N_v \) and \( M = l_h N_h \). Please note that the urban building façade’s model defined in Eq. (4.3) can be used even in cases where there is not any periodic structure in the given input façade we would like to model.

One toy example of a façade based on the model of Eq. (4.3) with:

\[
P_k = p_k p_k^t, \quad k = 1, 2, 3, \text{and}\]

\[
P_4 = P_1, \tag{4.4}
\]

where

\[
p_1 = \begin{bmatrix} 0_{1 \times 25} & 1_{1 \times 50} & 0_{1 \times 25} \end{bmatrix}^t
\]

\[
p_2 = \begin{bmatrix} 0_{1 \times 10} & 1_{1 \times 30} & 0_{1 \times 20} & 1_{1 \times 30} & 0_{1 \times 10} \end{bmatrix}^t
\]

\[
p_3 = \begin{bmatrix} 0_{1 \times 35} & 1_{1 \times 30} & 0_{1 \times 35} \end{bmatrix}^t
\]

\[
M_1 = \begin{bmatrix} 1_{3 \times 2} & 0_{3 \times 1} \\
0_{2 \times 2} & 0_{2 \times 1} \end{bmatrix}, \quad M_2 = \begin{bmatrix} 0_{3 \times 2} & 0_{3 \times 1} \\
1_{2 \times 2} & 0_{2 \times 1} \end{bmatrix}
\]

\[
M_3 = \begin{bmatrix} 0_{3 \times 2} & 1_{3 \times 1} \\
0_{2 \times 2} & 0_{2 \times 1} \end{bmatrix}, \quad M_4 = \begin{bmatrix} 0_{3 \times 2} & 0_{3 \times 1} \\
0_{2 \times 2} & 1_{2 \times 1} \end{bmatrix}
\]

is shown in Figure [4.1(a)]. Note that the above defined matrices \( M_k, k = 1, 2, 3 \) satisfy Eq. (4.2).

In addition, as it is clear from Eq. (4.4-4.6), all matrices as well as all patterns are of rank one.
Generalizing Eq. (4.3) to permit a "wall" gray level $\lambda_0$, we get:

$$ F_{N \times M} = \lambda_0 1_N 1_M^t + \sum_{k=1}^{K} \lambda_k (M_k \otimes P_k). \quad (4.7) $$

Using the fact that the components of the partition of orthogonal array $1_{l_v \times l_h}$ of Eq. (4.2) are mutually exclusive, we rewrite Eq. (4.7) as:

$$ F_{N \times M} = \sum_{k=1}^{K} \lambda_k (M_k \otimes \hat{P}_k), \hat{P}_k = P_k + \frac{\lambda_0}{\lambda_k} 1_{N_v} 1_{N_h}^t \quad (4.8) $$

where $\hat{P}_k$ are modified patterns as defined above, and $1_{N_v} , 1_{N_h}$ are all ones vectors with the subscripts denoting their lengths.
CHAPTER 4. KRONECKER PRODUCT MODEL: THEORY

4.3 Recovering the Ideal Façade Model

In this section we would like to compute (or approximate) the components of the Kronecker product that generate a given ideal (i.e. noise-free) building façade $F_{N \times M} \in \mathbb{R}^{N \times M}$ with $N = l_v N_v$ and $M = l_h N_h$. Using the model defined in Eq. (4.8) we can define the following cost function:

$$C_F(M_k, \hat{P}_k, \lambda_k, k = 1, \cdots, K) = ||F_{N \times M} - F_{N \times M}||^2_2$$

$$= ||F_{N \times M} - \sum_{k=1}^{K} \lambda_k (M_k \otimes \hat{P}_k)||^2_2,$$

(4.9)

where $M_k$, $\hat{P}_k$ and $\lambda_k$, $k = 1, 2, \cdots, K$ denote the partition matrices, the patterns and the weighting factors of façade’s model respectively. As it is clear from its definition $C_F(.)$ is a Frobenious norm based cost function that quantifies the error between the given matrix $F_{N \times M}$ and the model $F_{N \times M}$.

Therefore, the modeling problem of the given urban building façade $F_{N \times M}$ can be expressed by the following minimization problem

$$\min_{M_k, \hat{P}_k, \lambda_k, k = 1, \cdots, K} C_F(M_k, \hat{P}_k, \lambda_k, k = 1, \cdots, K),$$

(4.10)

which is known as the Nearest Kronecker Product problem [Loan, 2000]. The following partition of the given matrix $F_{N \times M}$ is key for the solution of the above problem:

$$F_{N \times M} = \begin{bmatrix}
F_{11} & F_{12} & \cdots & F_{1l_h} \\
F_{21} & F_{22} & \cdots & F_{2l_h} \\
\vdots & \vdots & \ddots & \vdots \\
F_{l_v 1} & F_{l_v 2} & \cdots & F_{l_v l_h}
\end{bmatrix},$$

(4.11)

where $F_{ij}$ is a block of size $N_v \times N_h$. An illustration of the partition approach is shown in Figure 5.7 We can then form the matrix

$$\tilde{F}_{l_v l_h \times N_v N_h} = \begin{bmatrix}
vec(F_{11}) & vec(F_{21}) & \cdots & vec(F_{l_v 1}) & vec(F_{12}) & \cdots & vec(F_{l_v 2}) & \cdots & vec(F_{l_v l_h})
\end{bmatrix}^T$$

(4.12)
which constitutes a rearrangement of the given façade matrix $F_{N \times M}$. An example of $\tilde{F}_{l_v h \times N v N_h}$ is illustrated in Figure 5.4. Using the above defined quantities, the cost function of Eq. (4.9) can be equivalently expressed as:

$$C_F(m_k, \hat{p}_k, \lambda_k, k = 1, \cdots, K) = \|\tilde{F}_{l_v h \times N v N_h} - \sum_{k=1}^{K} \lambda_k m_k \hat{p}_k^T\|_2^2$$  \hspace{1cm} (4.13)$$

where $m_k, \hat{p}_k$ are the column-wise vectorized forms of matrices $M_k, \hat{P}_k$. By exploiting the above defined equivalent form of the cost function, the Kronecker Product SVD \cite{Loan_2000} can be used to solve the optimization problem of Eq. (4.10):

**Theorem 1:** Let $\tilde{F}_{l_v h \times N v N_h} = V \Sigma U^T$ be the Singular Value Decomposition of the rearranged counterpart of matrix $F_{N \times M}$. Let us also consider the following diagonal matrix

$$\Sigma_K = \text{diag} \{\sigma_1 \sigma_2 \cdots \sigma_K\}$$  \hspace{1cm} (4.14)$$

containing the first $K$ singular values of matrix $\tilde{F}_{l_v h \times N v N_h}$, and let

$$V_K = [v_1 v_2 \cdots v_K] \hspace{0.5cm} U_K = [u_1 u_2 \cdots u_K]$$  \hspace{1cm} (4.15)$$

be the $K$ associated left and right singular vectors respectively. Then, the matrices $M_k^*$, the patterns $\hat{P}_k^*$, and the weighting factors $\lambda_k^*$ that satisfy:

$$\text{vec}(M_k^*) = v_k, \text{vec}(\hat{P}_k^*) = u_k, \lambda_k^* = \sigma_k, \hspace{0.5cm} k = 1, 2, \cdots, K$$  \hspace{1cm} (4.16)$$

constitute the optimal solution of the optimization problem of Eq. (4.10).

Using Theorem 1, we can find an optimal approximation that has the desired form, i.e. it is a sum of Kronecker products, that minimizes the cost function defined in Eq. (4.9). Note, however, that some of the characteristics of the optimal solution, are not consistent with the ingredients of the façade model defined in (4.7) thus making the direct use of Theorem 1 problematic. Specifically,
neither the optimal matrices $M_k^\star$ nor the optimal patterns $\hat{P}_k^\star$ have, in the general case, the desired form, that is they are not 1-0 matrices and piecewise constant surfaces, respectively. In addition, the vectorized form of the optimal patterns are orthonormal to each other.

In order to impose one of the requirements of the proposed façade model, in the sequel we consider that matrices $M_k$ have the desired 1−0 form and are known. In such a case, we form the cost function:

$$\hat{C}_F(\hat{P}_k, \lambda_k, k = 1, \cdots, K | M_k),$$

(4.17)

which is the cost function of Eq. (4.10) but with the partition matrices known. We would like to minimize it with respect to the patterns $\hat{P}_k$ and the weighting factors $\lambda_k$. The solution of the new optimization problem is the subject of the next lemma.

**Lemma 1:** Assuming that the matrices $M_k$, $k = 1, 2, \cdots, K$ defined in Eqs. (4.1-4.2) are known, then the minimization of the cost function defined in Eq. (4.17) produces patterns $\hat{P}_k$ and weighting factors $\lambda_k$ that are related as follows:

$$\lambda_k^\star vec\{\hat{P}_k^\star\} = \frac{U\Sigma^T V^T vec\{M_k\}}{||vec\{M_k\}||_2^2}, k = 1, 2, \cdots, K$$

(4.18)

**Proof:** Using the fact that $||F||_2^2 = \text{trace}\{F^TF\}$, the SVD decomposition of the rearranged counterpart of matrix $F_{N \times M}$, the linearity of the trace operator, and after some simple mathematical manipulations, the cost function defined in Eq. (4.13) can be rewritten as follows:

$$C_F(\hat{P}_k, \lambda_k, k = 1, \cdots, K | M_k) = \text{trace}\{U\Sigma^T U^T\} - \text{trace} \left\{U\Sigma^T V^T \sum_{k=1}^{K} \lambda_k m_k \hat{p}_k^T + \sum_{k=1}^{K} \lambda_k \hat{p}_k m_k^T V \Sigma U^T\right\} + \text{trace} \left\{\left(\sum_{k=1}^{K} \lambda_k m_k \hat{p}_k^T\right) \left(\sum_{k=1}^{K} \lambda_k \hat{p}_k m_k^T\right)\right\}.$$

Moreover, using the orthogonality of vectors $m_k, k = 1, 2, \cdots, K$, the orthonormality of matrix $U$, the commutative property of trace operator, and by interchanging the order of summations and
trace operator, we obtain:

\[
C_F(\hat{\mathbf{p}}_k, \lambda_k, k = 1, \cdots, K \mid \mathbf{M}_k) = \text{trace}\{\Sigma^T\Sigma\} - 2 \sum_{k=1}^{K} \lambda_k \text{trace}\{\hat{\mathbf{p}}_k^T \mathbf{U} \Sigma^T \mathbf{V} \mathbf{m}_k\} + \sum_{k=1}^{K} \lambda_k^2 \|\mathbf{m}_k\|_2^2 \|\hat{\mathbf{p}}_k\|_2^2.
\]

By taking the partial derivatives of the above function with respect to all components of the parameters \(\hat{\mathbf{p}}_k, k = 1, 2, \cdots, K\), stacking and setting them to zero we obtain the desired result.

Note that if we substitute into (4.18) the optimal solution of Eq. (4.16) for \(\mathbf{M}_k\), the optimal solution of the patterns as well as the weighting factors coincide with those in Eq. (4.16) as they owed to be. Note also that according to Eq. (4.18), the vectorized forms of the optimal patterns are not necessarily orthonormal to each other, unlike Theorem 1.

We applied both of the above optimal solutions for the modeling of urban building shown in Figure 5.1(left) and the resulting rank 4 solutions are shown in Figures 5.1(middle) and 5.1(right). The matrices \(\mathbf{M}_k, k = 1, 2, 3, 4\) we used for the evaluation of the optimal solution of Eq. (4.18) are:

\[
\mathbf{M}_1 = \begin{bmatrix} 1_{2 \times 3} & 0_{2 \times 7} \\ 0_{1 \times 3} & 1_{1 \times 7} \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} 0_{2 \times 3} & 0_{2 \times 7} \\ 1_{1 \times 3} & 0_{1 \times 7} \end{bmatrix},
\]

\[
\mathbf{M}_3 = \begin{bmatrix} 0_{3 \times 3} & 1_{3 \times 1} & 0_{3 \times 6} \\ 0_{3 \times 4} & 1_{3 \times 6} \end{bmatrix}, \quad \mathbf{M}_4 = \begin{bmatrix} 0_{3 \times 3} & 0_{3 \times 6} \\ 1_{3 \times 6} \end{bmatrix}.
\]

It is evident from Figures 5.1(middle) and 5.1(right) that the optimal solution resulting from the application of Eq. (4.18) outperforms the former one as expected.

4.4 Discussion

Lemma 1 is a powerful tool that can be used for solving the modeling problem of urban building façades. However, its use demands knowledge on the partitioning 1−0 matrices \(\mathbf{M}_k, k = 1, \ldots, K\). In real façade images, all the above mentioned urban building façade models constitute idealizations of the real ones. From this point of view singular vectors may lack any meaning in terms of the properties of the data. For example, it is well known that singular vectors play a very important role in the Karhunen Loeve (KL) transform when the data are drawn from a Gaussian distribution, and rank -K solution offers the optimal solution when the given matrix is corrupted by i.i.d Gaussian
noise. The major drawback of PCA is its sensitivity to errors of large magnitude even if matrix is contaminated with such errors in a very small part. In fact, a single corrupted entry can throw the resulting low-rank matrix arbitrarily far from the true solution. In the next chapter, inspired by Eq. (4.18), we present a clustering based technique to estimate the spacial periods for the partition purpose and finally solve the Kronecker approximation problem.
Chapter 5

Applying the Kronecker Product Model to Repeated Patterns Detection

5.1 Introduction

In this chapter, we will illustrate the implementation of algorithms that aim to detect repetitive patterns by solving for the variables in the Kronecker Product model defined by Eq. 4.8. Repetitive patterns or periodic structures detection has received significant attention in both 2D images [Zhao et al., 2010] and [Teboul et al., 2011b] and 3D point clouds [Friedman and Stamos, 2013] and [Shen et al., 2011]. Repeated patterns are usually hypothesized from the matching of local image features. They can be modeled as a set of sparse repeated features [Schindler et al., 2008a] in which the crystallographic group theory [Liu et al., 2004a] was employed. The work of [Wu et al., 2010a] maximizes local symmetries and separates different repetition groups via evaluation of the local repetition quality conditionally for different repetition intervals.

The work of [Muller et al., 2007a] proposes an approach to detect symmetric structures in a rectified frontal-façade and to reconstruct a 3D geometric model. The work of [Yang et al., 2012a] describes a method for periodic structure detection upon the pixel-classification results of a rectified
façade. Shape grammars have also been used for 2D façade parsing [Teboul et al., 2011b]. Other similar grammar-based approaches include [Barinova et al., 2010].

5.2 Algorithm

Given a frontal façade image $F_{N \times M}$, most of the well known low-rank modeling techniques use the original image and try to minimize its rank. We, on the other hand, use a Kronecker product based façade model $\mathcal{F}_{N \times M}$, which is defined in 4.3 as follows:

$$\mathcal{F}_{N \times M} = \sum_{k=1}^{K} \lambda_k (M_k \otimes P_k).$$

We are thus able to express the cost function defined in Eq. (4.9) in an equivalent form (4.13). This is essential, since by transforming the given matrix $F_{N \times M}$ into its rearranged counterpart $\tilde{F}_{l_v l_h \times N_v N_h}$, we form a matrix whose rank is drastically reduced (it is upper bounded by the smallest dimension of the above mentioned matrix, which usually is equal to $l_v l_h$). Our algorithm starts with the estimation of the size $N_v \times N_h$ of the patterns (Section 5.2.1), continues with the estimation of $K$ and the actual partition matrices (Secs. 5.2.1.1-5.2.3) and concludes with the computation of pattern matrices and weights (Section 5.2.4).

5.2.1 Estimating the Spatial Periods of the Patterns

In this section we provide an algorithm that estimates the spatial period of the unknown patterns.

Although well known methods ([Friedman and Stamos, 2013] and [Shen et al., 2011]) can be used for that purpose, we adopt a k-means based algorithm proposed by [Liu et al., 2013b] to address this problem in an efficient way. Before we describe the spatial periods estimation algorithm, let us introduce this matrix rank estimation algorithm which is essential for estimating the spatial periods in Section 5.2.1.2.
Figure 5.1: A real building façade (top), its optimal modeling of rank 4 (middle), obtained from the solution of the optimization problem of Eq. (4.10), and its optimal modeling of rank 4 (bottom), obtained from the minimization of the cost function of Eq. (4.17) with matrices $M_k$ predefined (please see text).
CHAPTER 5. APPLYING THE KROENECKER PRODUCT MODEL TO REPEATED PATTERNS DETECTION

5.2.1.1 Estimating the Matrix Rank $K$ by Clustering

Here we will illustrate an iterative technique which is based on the idea of clustering the rows of a given matrix $F_{N \times M}$ [Liu et al., 2013b]. In particular, we use a partitional ($k$-means) clustering algorithm in an iterative fashion in order to accurately estimate the rank of that matrix. This algorithm is general for variable matrices and 2D images. In Section 5.2.2 we will exploit an alternative method that aims to estimate the matrix rank.

To this end, let us consider that matrix $F_{N \times M}$ (for simplicity in the notation from now on we will denote it by $F$), as well as the desired number of clusters we would like to group the rows of the matrix (let us denote it by $K$) be given, and let us define the following set consisting of $K$ groups:

$$
\mathcal{R}_k = \{f^t_q : \|f^t_q - \bar{r}^t_k\|^2_2 \leq \|f^t_q - \bar{r}^t_l\|^2_2, \forall 1 \leq l \leq K\} \\
= 1, 2, \ldots, K
$$

(5.1)

where $f^t_q$ denotes the $q$-th row of matrix $F$, and $\bar{r}^t_k$ the mean of the $k$-th group of the rows respectively, as computed by $k$-means.

Let us also define the corresponding indicator vectors of length $l_v l_h$ each:

$$
1_{\mathcal{R}_k}[q] = \begin{cases} 
1 & \text{if } f^t_q \in \mathcal{R}_k \\
0 & \text{otherwise}, \quad q = \{1, 2, \ldots, l_v l_h\}
\end{cases}
$$

(5.2)

and the element-wise mean vectors of each group:

$$
\bar{r}^t_k = mean\{\mathcal{R}_k\}, \quad k = 1, 2, \ldots, K
$$

(5.3)

We can now define the following matrix:

$$
F_R = \sum_{k=1}^{K} 1_{\mathcal{R}_k} \bar{r}^t_k
$$

(5.4)

which has the same size as $F$. More importantly, if the given number of clusters $K$ were the correct one, then $K$ should equal to the rank of $F$. If, on the other hand, the given number of clusters $K$
CHAPTER 5. APPLYING THE KRONECKER PRODUCT MODEL TO REPEATED PATTERNS DETECTION

is greater than the real rank of \( F \), then the rank of \( F_R \) will be smaller than \( K \). Hence, by defining the new number of the clusters as:

\[
K = \text{rank}(F_R) \tag{5.5}
\]

and repeating the above described procedure, we are expecting that after some iterations, \( F_R \) will be the desired approximation of \( F \). Note that the computation of rank in Eq. (5.5) and as part of Algorithm 3 is a generic algorithm and not one that minimizes the rank of a matrix.

Algorithm 3: Kronecker Façade Modeling, noise-free ideal case. Input: \( F, K = \text{rank}(F) \)

1: repeat
2: Form groups \( \mathcal{R}_k, k = 1, \ldots, K \) via \( k \)-means (5.1)
3: Form the indicator vectors \( 1_{\mathcal{R}_k} \) of (5.2)
4: Form the mean vectors \( \bar{r}_k \) of (5.3)
5: Compute the matrix \( F_R \) defined in (5.4)
6: Compute its rank \( K \) (5.5)
7: Assign \( F_R \) to \( F \)
8: until convergence
9: Output: \( F_R^*, K^*, 1_{\mathcal{R}_k} \).

Note also that the use of mean in Eq. (5.3) is in exact accordance with Lemma 1 (as will be seen in Section 5.2.4). This will provide the optimal result assuming an ideal noise-free case.

In practice though, due to variations caused by occlusions (such as trees, traffic lights, etc.), shadows, etc., instead of the mean in Step 4, we use the element-wise median operator:

\[
\bar{r}_k^t = \text{median}\{\mathcal{R}_k\}. \tag{5.6}
\]

This is based on the robustness of the median operator (used for the estimation of the most characteristic values of rows that belong to the same cluster) and its optimality in the \( L_1 \) sense.

A second modification is also essential. Unfortunately, Lemma 1 does not guarantee that the patterns are piece-wise constant. One way to enforce that constraint is by also forcing clustering in the columns of \( F \) as well (note that each column spans all patterns). We thus consider the matrix:

\[
G = \frac{1}{2}(F_C + F_R) \tag{5.7}
\]
and the new number of the clusters:

$$K = \min \{\text{rank}(F_R), \text{rank}(F_C)\},$$  \hspace{1cm} (5.8)

where $F_C$ is the column-wise clustering result. It is obtained by following the same $k$-means clustering, but now in the columns:

$$C_k = \{f_p : ||f_p - \bar{c}_k||_2^2 \leq ||f_p - \bar{c}_l||_2^2, \forall 1 \leq l \leq K\}$$

$$k = 1, 2, \ldots, K \tag{5.9}$$

where $f_p$ denotes the $p$-th column of matrix $F$, and $\bar{c}_k$ denotes the mean of the $k$-th group of the columns respectively. The corresponding indicator vectors of length $N_v N_h$ is defined as:

$$1_{C_k[p]} = \begin{cases} 
1, & \text{if } f_p \in C_k \\
0, & \text{otherwise}, \quad p = \{1, 2, \ldots, N_v N_h\}
\end{cases} \tag{5.10}$$

and the element-wise median vectors of each group:

$$\bar{c}_k = \text{median}\{C_k\}, \quad k = 1, 2, \ldots, K. \tag{5.11}$$

Then,

$$F_C = \sum_{k=1}^{K} \bar{c}_k 1_{C_k}^t. \tag{5.12}$$

Therefore, the algorithm we use in practice is shown below.
Algorithm 4: Kronecker Façade Modeling. Input: $F$, $K = \text{rank}(F)$

1: repeat
2: Form groups $R_k, C_k$ $k = 1, \ldots, K$ via $k$-means (5.1), (5.9)
3: Form the indicator vectors $1_{R_k}, 1_{C_k}$ of (5.2), (5.10)
4: Form the vectors $r_{k}, c_{k}$ of (5.6), (5.11)
5: Form the matrices $F_{R}, F_{C}$ and $G$ of (5.4), (5.12) and (5.7)
6: Set $K$ using (5.8)
7: Assign $G$ to $F$
8: until convergence
9: Output: $F^*, K^*, 1_{R_k}$.

Note that $1_{R_k} \star, k = 1, 2, \cdots, K^*$ are the rows of $F^*_R$.

For the convergence condition in Algorithm 4 we can consider the convergence of the sum of all entries in $|G_i - G_{i-1}|$, where $G_i$ denotes the $G$ obtained in the $i$th iteration, or set the number of iterations to a maximum pre-specified number. Finally, the denoising of matrix $F$ after Step 7 in the algorithm above, can drastically speed up the convergence.

5.2.1.2 Estimating the Spatial Periods

In this section, we will estimate the spatial periods by using the algorithms described in Section 5.2.1.1. Let us run Algorithm 4 for a predefined value $K_0$ of the parameter $K$ once with input $F$, and then with input $F^t$. Then, we can compute the following:

$$\|1_{R_k}\|_0 = \max_{k=1,2,\cdots,K_0} \{\|1_{R_k}\|_0\} \tag{5.13}$$

$$\|1_{C_i}\|_0 = \max_{l=1,2,\cdots,K_0} \{\|1_{C_i}\|_0\} \tag{5.14}$$

and the corresponding auto-correlation sequences:

$$r_{R_k} = 1_{R_k} \star 1_{R_k} \tag{5.15}$$

$$c_{C_i} = 1_{C_i} \star 1_{C_i} \tag{5.16}$$

where “$\star$” denotes the correlation operator. Note that by taking into account Eqs. (5.13,5.14), indicator vectors $1_{R_k}, 1_{C_i}$ are the vectors that define the dominant row and column spatial periods.
Figure 5.2: Estimation of the spatial periods of façade shown in Figure 5.1 (left). (a) Cross-Correlation sequences used for the estimation of $N_v = 90$ pixels and (b) estimation of $N_h = 56$ pixels. Please enlarge the image to see the coordinates. Distance between the adjacent peaks provides the period information.

respectively and thus the computation of the corresponding auto correlation sequences makes sense.

Note also that the vectors involved in the computation of the proposed auto-correlation sequences are based on indicator vectors, that is $1-0$ vectors, and not on gray-value quantities.

Algorithm 5: Estimation of Periods $N_h$, $N_v$.

Input: $F_{N \times M}$, $K_0$

1: Form the vectors $1_{R_k}, k = 1, 2, ..., K_0$ using (5.2)
2: Form the vectors $1_{C_l}, l = 1, 2, ..., K_0$
3: Compute the quantities defined in Eqs. (5.13-5.14)
4: Compute the sequences defined in Eqs. (5.15-5.16)
5: Use them to estimate the desired spatial periods
6: Output: $\hat{N}_h$ and $\hat{N}_v$.

The results we obtained with $K_0 = 5$ in the urban building façade of Figure 5.1 (left), are shown in Figures 5.2 (a) and 5.2 (b) respectively.

5.2.2 Estimating $K$ by Unbiased Estimator of the Degrees of Freedom

Here we already know the parameters $N_v$ and $N_h$ that are computed in Section 5.2.1. Recall that the given building façade $F_{N \times M}$ is partitioned into $l_v \times l_h$ blocks. Each block is then re-arranged into a vector, and $F_{N \times M}$ can be re-arranged in the form of Eq. (4.12).

In the original definition of the Kronecker Product Model in Eq. (4.3), an essential parameter $K$ to be estimated denotes the number of unique patterns among the partitioned blocks. If the given façade contains repeated patterns, then the partition blocks can be clustered into groups.
Each group is formed by the repetition of one pattern. Thus $K$ can be interpreted as the number of these groups. As we have re-arranged the partitioned blocks into vectors, each group of repeated patterns is in the form of a group of repeated vectors in $\tilde{F}$. Thus $K$ represents the rank of $\tilde{F}$, and please see Figure 5.5 for an example. Now the problem reduces to the estimation of rank of $\tilde{F}$. For simplicity, we will use $F$ to represent $\tilde{F}$ in this section. Then the problem can be formed as the following statistical problem of finding the correct rank of a perturbed low-rank matrix: given a noisy observation $F = F^o + E$, the goal is to estimate an $m_1 \times m_2$ matrix $F^o$, where rank$(F^o) = K$. We assume that the noise matrix $E$ follows a matrix norm distribution $N(0, \tau^2 I_{m_1} \otimes I_{m_2})$. We also assume that $m_1 \leq m_2$ so that the full rank is $m_1$.

A k-means clustering based iterative algorithm was used in [Liu et al., 2013c] to estimate the rank of $F$. However, the k-means based approach is computationally expensive and unstable in cases where there is occlusion caused by illumination or shadows. In [Yuan, 2011], an approach was proposed to address this problem via Degrees of Freedom estimators in an efficient and reliable way. Based on this idea, we will describe a statistical technique that estimates the rank $K$ of $F$.

In [Ye, 1998] and [Efron, 2004], a rigorous definition of degrees of freedom in the framework of Stein’s Unbiased Risk Estimate (SURE) was provided. For the classical linear regression, degrees of freedom is often associated with the number of variables in the model. However the parallel interpretation is unclear in the context of low-rank matrix estimation problems where the estimators are highly nonlinear in nature. The number of free parameters in specifying a low-rank matrix is often used as the degrees of freedom in this case. It was shown via both theory and numerical studies in [Yuan, 2011], that the number of free parameters incorrectly measures the complexity of the rank constrained estimator.

Let $F = U\Sigma V^\top$ be the singular value decomposition (SVD) of $F$, where $\Sigma$ is a diagonal matrix with diagonal entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{m_1} \geq 0$, and $U, V$ are orthonormal matrices. The estimator of $F$ with rank $K$, denoted as $\hat{F}_K$, is defined as:

$$\hat{F}_K = \sum_{k=1}^{K} \sigma_k u_k v_k^\top,$$

(5.17)
where $u_k$ and $v_k$ are the $k$th columns of $U$ and $V$ respectively.

The formal definition of the optimization function for estimating $K$ is formulated as:

$$
\ell(\hat{F}_K) = \|\hat{F}_K - F^0\|_F^2 \\
= \|\hat{F}_K - (F - E)\|_F^2 \\
= \|\hat{F}_K - F\|_F^2 + 2\times\langle\hat{F}_K - F, E\rangle + \|E\|_F^2 \\
= \|\hat{F}_K - F\|_F^2 + 2\times\langle\hat{F}_K, E\rangle + (\text{terms not depending on } \hat{F}_K),
$$

(5.18)

where $K \in \{1, \ldots, m_1\}$ is a tuning parameter, $\| \cdot \|_F$ stands for the usual matrix Frobenious norm, and $\langle \cdot, \cdot \rangle$ stands for the inner product. The first term measures the goodness of fit of $\hat{F}_K$ to the observation $F$. The second term can be interpreted as the cost of the estimating procedure and can be estimated by using degrees of freedom as shown in [Yuan, 2011].

$$
df(\hat{F}_K) = \frac{1}{\tau^2} \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \text{cov}(\hat{F}_{Kij}, E_{ij}).
$$

(5.19)

We refer the interested readers to [Ye, 1998] and [Efron, 2004] for further discussion about the general theory regarding degrees of freedom. Once the degrees of freedom are defined, the rank estimator can be constructed by using the following $C_p$ type statistic to select the proper rank $K$:

$$
C_p(\hat{F}_K) = \|\hat{F}_K - F\|_F^2 + 2\tau^2 \df(\hat{F}_K),
$$

(5.20)

where $\tau^2$ is defined as $\text{var}(\hat{F}_K - F)$.

Usually, the degrees of freedom defined in Eq. (5.19) are not directly computable. The unbiased estimator proposed in [Stein, 1981] is lack of analytical expressions and requires numerical methods such as data perturbation and re-sampling techniques which are computationally prohibitive in large scale problems.

For our specific rank regularized estimation problem, we employ the following unbiased estima-
tor of degrees of freedom (see details in Theorem 1 of [Yuan, 2011]).

\[
\hat{df}(\hat{F}_K) = (m_1 + m_2 - K)K + 2 \sum_{k=1}^{K} \sum_{l=K+1}^{m_1} \frac{\sigma_l^2}{\sigma_k^2 - \sigma_l^2},
\]

(5.21)

where \( \sigma_1 \geq \cdots \geq \sigma_K \geq \sigma_{K+1} \geq \cdots \geq \sigma_{m_1} \geq 0 \) are the singular values of \( F \). Using Eq. (5.21), we arrive at the following estimator of the \( C_p \) statistic for each candidate rank \( K \).

\[
\hat{C}_p(\hat{F}_K) = \|\hat{F}_K - F\|^2_F + 2\tau^2 \hat{df}(\hat{F}_K),
\]

(5.22)

The estimated rank \( \hat{K} \) is then defined as follows.

\[
\hat{K} = \arg \min_{1 \leq K \leq m_1} \hat{C}_p(\hat{F}_K).
\]

(5.23)

An example of the quantity \( \hat{C}_p(\hat{F}_K) \) as a function of \( K \) is shown in Figure 5.3.

The whole algorithm is summarized as follows.

**Algorithm 6: Kronecker Façade Modeling, estimation of rank \( K \).** Input: \( F \) in size \( m_1 \times m_2 \), where we assume \( m_1 \leq m_2 \). \( F \) is the rearranged matrix \( \tilde{F} \) as defined in Eq. (4.12).

1. \( C_{\min} \leftarrow \infty, \text{rank} \leftarrow m_1 \)
2. Compute the SVD of \( F \), \( F = U \Sigma V^\top \)
3. for \( K = 1 \) to \( m_1 \) do
4. \( \hat{F}_K \leftarrow \sum_{k=1}^{K} \sigma_k u_k v_k^\top \)
5. \( f_1(K) \leftarrow \|\hat{F}_K - F\|^2_F \)
6. \( \hat{df}(K) \leftarrow (m_1 + m_2 - K)K + 2 \sum_{k=1}^{K} \sum_{l=K+1}^{m_1} \frac{\sigma_l^2}{\sigma_k^2 - \sigma_l^2} \)
7. \( \tau^2 \leftarrow \text{var}(\hat{F}_K^{\text{rank}}(K) - F) \)
8. \( C_p(K) \leftarrow f_1(K) + 2\tau^2 \hat{df}(K) \)
9. if \( C_p(K) < C_{\min} \) then
10. \( C_{\min} \leftarrow C_p(K), \text{rank} \leftarrow K \)
11. end if
12. end for
13. Output: \( \hat{F}_{\text{rank}}, \text{rank} \).
Figure 5.3: This figure shows the rank estimation result for a real façade building image shown in Figure 5.1 (top). (a) Illustration of the estimation function in Eq. (5.20), where the global minima comes up at index 4. The corresponding index indicates the desired rank for the re-arranged matrix $F$ shown in Figure 5.4. Please see Algorithm 6 for the estimation process. (b) The enlarged figure area inside the green box in (a), which shows the global minima in a clearer way. (c) Illustration of the first term of function in Eq. (5.21). (d) Illustration of the degrees of freedom function in Eq. (5.19).

Figure 5.4: Vectorization of all the $3 \times 10$ blocks in Figure 5.7 (b). This matrix is of full rank 30.

Figure 5.5: The low-rank component of matrix in Figure 5.4, where $K = 4$. The rank is estimated by Algorithm 6.
5.2.3 Estimating Matrices $M_k, k = 1, 2, \cdots, K^*$

We estimate matrices $M_k, k = 1, 2, \cdots, K^*$ by reshaping each one of the $K^*$ above mentioned indicator vectors into their nominal form, that is, in a rectangular array of size $l_h \times l_v$ each.

\begin{algorithm}
\caption{Estimation of Matrices $M_k$. Input: $1_{\mathcal{R}_k},K^*$}
\begin{algorithmic}[1]
\For{$k=1$ to $K^*$}
\State $m_k = 1_{\mathcal{R}_k}$
\State $M_k = \text{reshape}(m_k, l_h, l_v)$
\EndFor
\end{algorithmic}
\end{algorithm}

We must stress at this point that it is easy to validate that the vectorized forms of the estimated partition matrices satisfy the conditions of Eqs. (4.1) and (4.2).

5.2.4 Computing Patterns and Weighting Factors

At this point we have estimated all the quantities needed to find out the optimal patterns $\hat{P}_k$ and weighting factors $\lambda_k, k = 1, 2, \cdots, K^*$, as they are defined in Lemma 1. Note that the estimated partition matrices have the desired optimal $1 - 0$ form. In addition, since each $m_k$ coincides with the corresponding indicator vector, and by the definition of mean vectors $\bar{r}_k^*$ defined in (5.3), each term of the matrix $\tilde{F}_\mathcal{R}$ of (5.4), has exactly the same form with the optimal patterns defined in Lemma 1. Indeed, by taking into account that by definition $m_k = \text{vec}(M_k)$, and because of the special $1 - 0$ form of the partition matrices $\|m_k\|_2^2 = \|m_k\|_0$, the following is true:

$$\lambda_k^* \text{vec}(\hat{P}_k^*) = \frac{U\Sigma V^T \text{vec}(M_k)}{\|\text{vec}(M_k)\|_2} = \bar{r}_k^*, k = 1, \ldots, K^*. \tag{5.24}$$

Therefore, the vectors $\bar{r}_k^*$, computed in Algorithm 3 provide us the weighted optimal patterns. In practice, as discussed in Section 5.2.1.1 we are using the results of Algorithm 3.

5.2.5 Patterns Refinement

Our low-rank method (Sections 5.2.1.1 5.2.3) enables us to remove occlusions, small illumination variations and photometric distortions as seen in the fourth column of Figure 5.10 5.11 and 5.12.
CHAPTER 5. APPLYING THE KRONECKER PRODUCT MODEL TO REPEATED PATTERNS DETECTION

Figure 5.6: Refining the representative pattern for each group.

Figure 5.7: (a) Partitioning the façade into blocks by using the spatial periods estimated in Figure 5.2, where the green lines shows the partition blocks’ boundary. (b) Isolating the corresponding $3 \times 10$ blocks.

Because of this we have very accurate detection of repeated patterns. This can largely improve classification results. Based on those clean patterns, we can easily obtain 1-0 patterns (i.e. refining the results) by applying classification methods, such as the rank-one algorithm [Yang et al., 2012a], within each group. Examples of detected 1-0 patterns are shown in the last column of Figures 5.10, 5.11 and 5.12.

For example the method of [Yang et al., 2012a] fails in the case of Figure 5.11 due to tree occlusion. Our algorithm, however, can successfully detect four different clusters and clear pattern structures.

Figure 5.8: (a) Reshaping each row vector of matrix in Figure 5.5 to partition blocks. (b) Reconstructing façade image by pasting all blocks together.
CHAPTER 5. APPLYING THE KRONECKER PRODUCT MODEL TO REPEATED PATTERNS DETECTION

Figure 5.9: Grouping. Each color represents one group, and all patterns in a group have the same structure.

Figure 5.10: An overview of the repeated pattern detection procedure via Kronecker Product model. (a) Input image. (b) Partition grid showing the periods estimated by Algorithm 5, with all partition blocks colored randomly. (c) Grouped blocks generated by Algorithms 4 and 7, with each group having the same color. (d) Low-rank component generated by Algorithms 4 - 5 in Sections 5.2.1 - 5.2.1 (e) Estimated 1-0 repeated patterns by refining detection results shown in (d).
CHAPTER 5. APPLYING THE KRONECKER PRODUCT MODEL TO REPEATED PATTERNS DETECTION

Figure 5.11: From left to right: follow the order of Figure 5.10. The third row is a failure case of method presented in [Zhao and Quan, 2011] (large tree occlusion), but our method can successfully detect the repeated patterns as shown in the last column. The fifth row shows detected patterns for the example in Figure 5.1.
CHAPTER 5. APPLYING THE KRONECKER PRODUCT MODEL TO REPEATED PATTERNS DETECTION

Figure 5.12: From left to right: follow the order of Figure 5.10. First eight rows show success cases (robustness to occlusions and different architectural styles). Last row is a failure case due to the photometric variation and inability to model via a Kronecker product model.
5.3 Experiments and Discussion

The experiments are implemented in Matlab, and run on a computer with an 1.8 GHz Intel Core i7 CPU and a 4GB memory. To evaluate the performance of the Kronecker Product Model, we tested our repeated pattern detection in 95 images for which we had ground-truth [Teboul et al., 2011b; Yang et al., 2012a]. Out of the 95 images we tested, only 3.15% resulted in failure detection (see failure cases in Figure 5.12). The results from the remaining 96.85% were very similar to the ground-truth. We overlaid our results with the ground-truth pixel by pixel and had average matches for 93.01% of the pixels.

We can conclude that the block partition Section 5.2.1 is not a bottleneck of our algorithm. The partition lines may pass across the desired patterns, as shown in the second row of Figure 5.12. In such cases some pattern is divided into two adjacent partition blocks, such that the partition blocks don’t contain the desired patterns completely. However our algorithm is robust enough to detect them separately. We must stress at this point that a better partition can definitely improve the performance. In order to have partition lines mostly passing through wall areas as desired, we can adopt methods proposed in Muller et al., 2007a and Friedman and Stamos, 2013.

In the experiments, we found most of the common building façades to be able to be modeled by our Kronecker product structure. One limitation is that our method fails when a façade contains repeated structures that do not follow the Kronecker product model. Another limitation is the inability to handle large photometric variations, since they are causing ambiguity in the block partition (last row of Figure 5.12). Unfortunately, currently there is no simple way for the system to automatically determine failure cases.

5.4 Summary

In this chapter, we presented a novel method for detection of repeated patterns following a Kronecker Product formulation in the last chapter. Our method is general and can be applied to a wide variation of façade structures and is being based on a solid theoretical foundation. The fact that we are utilizing the low-rank part of the rearranged input façade image allows us to handle
problems of occlusion, shadows and illumination variations.
Chapter 6

Conclusion

6.1 Novelty and Contributions of Our Methods

This dissertation is dedicated to developing low-rank algorithms for urban images processing. We exploited novel methods for image rectification and repetitive patterns detection on urban images. We have demonstrated both in theory and practice with clear performance gains in a variety of experiments. Below we summarize our contributions:

- **Image rectification algorithms.** These methods aim to rectify the image precisely and automatically by combining traditional feature-based methods and low-rank techniques. These algorithms take advantages offered by vanishing points and low-rank invariant features, but avoid their weakness.

- **Automatic façade texture extraction algorithms.** These methods automatically detect façade regions in 2D urban images using vanishing lines and distribution of Harris corners. The rectified frontal façades are generated for a variety of fundamental tasks, such as façade parsing, repetitive patterns detection and 3D reconstruction.

- **Repetitive patterns detection algorithms on frontal façades.** We are the first to propose the novel Kronecker Product model, which integrates low-rank textures, robust PCA and the Kronecker product, for modeling a frontal façade structure globally in an efficient and effective
way. This model is built upon a solid theoretical foundation and has been demonstrated by sufficient experiments. The input of this method is a frontal façade texture, which can be generated by the approach described in Chapter 3. The proposed algorithms are able to simultaneously detect repeated patterns, recover façade structures and remove noise and occlusions. The Kronecker Product model is general and can be applied to a wide variation of façade structures. The fact that we are utilizing the low-rank component of the rearranged input façade image allows us to handle the problem of occlusion, shadows and illumination variation.

6.2 Limitations and Future Direction

6.2.1 Automatic Texture Selection and Façade Segmentation

In our experiments for automatic texture selection algorithm, both of the parameters $m$ and $n$ in drawing vanishing lines from vanishing points towards the façade planes were associated with a fixed value 10, which somehow limited the performance of the façade texture quadrilateral selection. Most of the urban images in the existing image databases, such as TILT database and our database, contain dominant façade planes, and thus the automatic texture selection algorithm in Chapter 3 works well. For aerial images that contain hundreds of building façades, our algorithm will be affected due to the fact that buildings in aerial images generally are of low resolution and that the 10×10 partition blocks may contain multiple façades. The situation becomes worse when buildings are not aligned or are built in random directions.

In this case, one can utilize object segmentation algorithms for extracting rough façade regions. This strategy can largely narrow down the searching range of façade textures. Based on the rough façade regions, it will be much easier to obtain low-rank façade textures. For instance, an automatic regularity-driven framework proposed by [Liu and Liu, 2014] can detect hundreds of façades from aerial images of urban scenes. This method can handle images that have wide viewing angles and contain more than 200 façades per image. The detected façade regions can be utilized as input of our urban low-rank algorithm, which can easily recover the perspective transformation. The
rectified façades are further used as input of the Kronecker Product model, and repetitive patterns for all detected façades will be generated. For ground-level imagery that includes multiple façades, one can use the methods proposed by Zhao et al., 2010, Hurley and Rickard, 2009, Recky et al., 2011, Schindler et al., 2008b and Wendel et al., 2010.

6.2.2 Nested Patterns

A nested pattern can be defined as a repeated texture which itself includes smaller indivisible patterns like doors and windows (see the red box in Figure 6.1(a) for an example). In the current Kronecker Product model, the façade is partitioned into small blocks containing either one single window or the façade wall. We can view the whole façade as a Kronecker Product of such nested patterns, where each nested pattern contains three indivisible windows (Figure 6.1(b)). Each nested pattern (see the yellow box and red box in Figure 6.1(b)) can be further modeled as a Kronecker Product of three single windows.

In the current Kronecker Product model, the façade is segmented into the smallest possible patches, where each patch contains either one single window or the façade wall. The essential reason for the block partition comes from the period computation described in Section 5.2.4, where a Cross-Correlation function based algorithm is exploited to compute the most frequently appearing period. In order to compute the period along horizontal direction for the façade in Figure 6.1, the current Kronecker Product model computes the distance between closest adjacent peaks of the cross-correlation function as shown in Figure 6.2 top, thus an input façade is always split into the smallest patterns. This method works well for smaller façades, where the illumination variance is not large enough. When considering skyscrapers, the top part in the façade generally contains stronger illumination, where the specularly reflected light from glass windows and doors makes the repeated patterns much brighter than their surrounding wall areas, while in the bottom part of the building façades, the patterns are much darker due to shadows from the neighboring buildings. The situation becomes worse when the images are in low resolution, where single patterns are vague. So if it is possible to find repeated textures that contains multiple patterns, then we can iteratively apply the Kronecker Product model to these nested textures.
Figure 6.1: A façade example that contains nested patterns. (a) The input image where the red box shows a nested pattern that contains three single windows. (b) By adopting nested patterns, the façade can be partitioned into repeated textures that contains multiple indivisible structures like doors and windows.

Figure 6.2: The Cross-Correlation function along horizontal direction for the façade in Figure 6.1. There actually are at least two types of repetition periods, denoted by the higher peaks and low peaks respectively. Top: in the current model, the distance between two adjacent peaks of the Cross-Correlation function is computed as the period. Bottom: we can find a strategy to isolate the higher peaks for the nested patterns. The façade can be partitioned into repeated textures that contains multiple indivisible structures as shown in Figure 6.1.
In order to solve this problem, we must find a way to compute a proper spatial period for façade partition. As shown in Figure 6.2 bottom, instead of finding the distance between the adjacent peaks, we can isolate the higher peaks where the distance between adjacent pairs denotes the period for bigger patterns. In this way, we can partition the façade into nested patterns and further detect the low-rank texture based on these nested patterns. This step removes part of noise in the façades and makes the smaller single patterns clearer. We can then further apply the Kronecker Product model to each type of these nested patterns to model the smaller indivisible patterns.

In some complicated situations, such as hundreds of different types of patterns contained in a façade, this method can then be exploited iteratively until all single patterns are modeled.

6.2.3 Types of Patterns

Our algorithm is able to detect repeated patterns and to cluster these patterns into different groups in an efficient and accurate way. However the types or labels of these patterns (doors, windows, balconies and shops) are not assigned by this method. Some existing methods [Teboul et al., 2013] and [Teboul et al., 2011a] solve the labeling problem via machine learning methods that define shape grammars for a particular type of buildings. It is a natural idea to combine the shape grammars and the Kronecker Product model to label these detected patterns.

6.2.4 3D Point Clouds

While it has been proved that low-rank based algorithms work well for 2D images, low-rank techniques are rarely applied to 3D point clouds processing. The 3D data acquisition can be extremely noisy due to illuminations, traffic lights, trees, and passing by pedestrians. An example is shown in Figure 6.3. In order to develop down streaming 3D modeling projects, denoising the data set becomes critical. In addition to noisy points, most of the cars are incomplete due to unwanted objects, such as trees, traffic lights, street signs, vehicles and pedestrians, in front of 3D scanners during the data acquisition process. Thus, in the future, we aim to develop low-rank based algorithms for automatically removing noisy points, completing car shapes and classifying these cars based on their types (SUVs, sedans or trucks), given a set of 3D car point clouds.
Figure 6.3: (a) A 3D point cloud of the city scene from the Wright State Ottawa dataset. (b) A 3D point cloud of a car. [Zelener et al., 2014]
Bibliography


[Candes et al., 2011] Emmanuel J. Candes, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? ACM, 2011.


[Zhao et al., 2010] Peng Zhao, Tian Fang, Jianxiong Xiao, Honghui Zhang, Qinping Zhao, and Long Quan. Rectilinear parsing of architecture in urban environment. CVPR, 2010.

