Video-based Face Alignment Using Efficient Sparse and Low-rank Approach

WU, King Keung

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Mechanical and Automation Engineering

The Chinese University of Hong Kong
August 2011

Thesis/Assessment Committee

Professor CHUNG Chi Kit Ronald (Chair)
Professor YAM Yeung (Thesis Supervisor)
Professor HUI Kin Chuen (Committee Member)
Professor KREINOVICH Vladik (External Examiner)

Abstract of thesis entitled:

Video-based Face Alignment Using Efficient Sparse and Low-rank Approach

Submitted by WU, King Keung

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in August 2011

The theme for this thesis is the application of latest sparse and low-rank techniques to face alignment problem. Face alignment is a challenging problem which has a wide variety of applications, for example in face recognition and photo-real talking head. This is because face shape is not a rigid but a deformable object, which allows various expressions, poses and sometimes very extreme illumination conditions.

Several methods have been introduced to tackle the alignment problem. Rather than using the popular Active Appearance Model (AAM), we follow a pixel-based approach called Robust Alignment by Sparse and Low-rank decomposition for linearly correlated images (RASL). It is a robust and highly accurate batch image alignment method which formulates the batch alignment problem as the solution of convex programs, with the aid of the latest advances in rank minimization.

RASL has been shown to work even under occlusions, corruptions, and illumination variations. Unfortunately, the origi-

nal formulation has limitation on scalability. The memory constraint as well as the computational cost restrain its applications on very large dataset. In this thesis, we investigate the performance of RASL on face alignment and then extend the original RASL to be applicable in large datasets. First we propose Mulit-RASL which serves as a direct extension of RASL by choosing a reference image and run RASL algorithm for multiple times. However, the performance is not as satisfactory as RASL in practice. Hence we introduce another approach by aligning the image in the dataset one-by-one. This one-by-one approach is based on two sparse concepts, sparse representations and sparse errors. They exploit the relationship between sparseness and l_1-norm minimization. Our proposed face alignment methods have been applied on the photo-real talking head, a challenging application which requires highly precise alignments of faces from video sequences. Experiments show that our one-by-one approach outperforms Multi-RASL for video-based face alignment.

In order to achieve our objective of efficient and accurate face alignment, the latest researches about sparse representations are reviewed and organized. This includes extensive studies of techniques of l_1-norm minimization. We develop the sparse representation techniques to deal with sparse corruption problem, where l_1-error is introduced based on l_1-norm minimization. Moreover, since our application requires fast computation, we derive our own algorithms for solving our designed l_1-norm minimization objectives by using split Bregman algorithm, which is one of the fastest l_1-algorithm. We verify our derived algorithms using

surveillance videos, showing that our proposed algorithms are in fact general, which can be applied to other applications in image processing other than face alignment, for instance, video segmentation problem.

本論文的主題是利用最新的稀疏和低秩的技術來解決人臉對齊的問題。人臉對齊是一個具有挑戰性的問題，它有各式各樣的應用，例如用於面部識別和利用真實照片建構的頭部模型上。因爲臉形可以變形，例如可以有各種表情、姿勢、有時甚至受燈光照明影響。

之前曾經有幾種方法被引入到解決對齊問題，例如主動外觀模型(AAM)。不過我們在這裡嘗試利用一個以像素爲基礎的方法，名爲RASL。它是針對線性相關圖像，利用最新的稀疏和低秩方法來實現魯棒對齊。他可以高度精確地以批量的方式對齊，將之轉化爲一個秩最小化問題。

RASL 已被證明可以對受阻礙、受損壞和受光照影響的情況進行對齊。可惜受限於記憶體太少和低運算力，它不適用於數據量大的情況。本論文會將 RASL 應用到人臉對齊的問題上，然後將 RASL 推廣到應付數據量大的情況。首先，我們提出一個通過選取一個參考圖像和重複執行 RASL 算法，直接將 RASL 推廣的方法，命名過 Multi-RASL。不過這方法的效能並不及原來的 RASL。因此，我們提出另一個方法，以逐一地對齊人臉取代一次過對齊全部人臉。這個方法建基於兩個概念：稀疏表示和稀疏誤差，其中我們會利用稀疏性和 l_1-norm 最小化問題。我們提出去人臉對齊方法特別針對真實照片建構的頭部模型應用上，因爲它要求高度精準地對一段或多段影片中的人臉對齊。實驗証明我們這方法比 Multi-RASL 在這個應用上優勝。

爲了達到有效而準確地對齊人臉的目標，我們將最新的稀疏表示研究作了概括和整理，特別是包括 l_1-norm 最小化問題。其中，我們提出 l_1-error 來應付稀疏誤差的問題。由於我們的應用要求快速運算，故此選了其中一種現時最快的 l_1 算法：split Bregman 算法。我們用監控錄像來驗証我們的算法無誤，而且這些算法不單可用在人臉對齊，還可用到其他圖像處理的應用上，例如視頻分割的問題等。

Acknowledgement

There are many people who helped me in my Master study, and I would like to take this opportunity to thank them.

First and foremost, I would like to express my most sincere gratitude to my thesis supervisor Professor Yeung Yam for giving me lots of opportunity to learn and improve myself, in both academic and personal skills. I am glad to have a supervisor who gave me the freedom to choose whatever I would like to explore and also the guidance to steer me to a different directions when I reached a dead end. Under his supervision, I learn how to tackle difficult problems and also how to appreciate and enjoy doing research.

During my study period, I am fortunate to be given the chance to an internship in Microsoft Research Asia (MSRA), where I met many great researchers in the field of visual computing and speech. I would like to thank my mentors in Speech Group of MSRA, Dr. Frank Soong and Dr. Lijuan Wang, for their helpful insights. They gave me lots of inspirations and support. Besides, I also thank John Wright in Visual Computing Group for teaching me the essence of compressed sensing and sparse representations.

I would like to thank all my current and former colleagues

in ICSL: Josh Lam, S. M. Wong, Coco Ho, K. K. So, Melvin Cheung, Adam Li, Derek Yu and Heran Song. They provide me a peaceful place to do research. Whenever I need any help, they would give me enough support. I am grateful to have them to be my colleagues.

Special thanks to Professor M. C. Chu from Physics department, who helps me a lot to transfer from physics to engineering (and invite me to play basketball with him every Friday). Besides, thanks very much to Professor Helen Meng for her encouragement to participate in PhD Forum, even though I am still not yet a PhD student.

Finally, I would like to thank my family. Thanks mum and dad for their love and financial support throughout my life. More than 20 years have been passed since kindergarten. I hope they (and also my younger brother Victor, who is graduating his Bachelor degree this year) can share my happiness of graduation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Face Alignment Algorithms

Face alignment is the main theme of this thesis. It is difficult yet very important in many applications such as robust face recognition [46] and photo-realistic talking head [41]. This section gives a summary to the past research on this challenging problem.

1.1.1 Objectives

To see how important and useful face alignment is, it is better to illustrate with a practical example. In face recognition, usually several photos of a person are taken to compare with a face dataset so as to identify the individual. However in practice, the test photos taken will not be exactly in the same orientations as those in the dataset, for example the person is not exactly in the frontal normal view with respect to the camera, or his/her head is tilted with a small angle. Although those variations can be small in terms of angles, the comparison result can vary a lot and eventually causing the recognizer to fail [36, 42]. Therefore,

face alignment is necessary for face recognition.

The objective of face alignment is to localize facial features such as eyes, mouth, nose, and eye-brows, etc. It is an example of image registration. Sometimes, a specific pose is given so that all the face images are aligned to it through deformation and warping. It can be considered as a special case of the general image alignment problem. However, it is worth to notice that not all the image alignment methods can be directly applied on the face images since human face is a deformable object rather than a rigid one. Variations in expressions and poses cause the alignment becomes a great challenge. Besides, the possibility of extreme illumination conditions also poses a huge difficulty to the problem. As a result, research of face alignment have been an active research field over the recent decade.

1.1.2 Motivation: Photo-realistic Talking Head

Before discussing the algorithms, let us state the motivation of our project first. Our interest in face alignment begins with the application of photo-realistic talking head. Photo-realistic talking heads have a wide variety of applications in human-machine interaction, from entertaining purpose in video games to educational software assisting language learning. A vividly lip-sync talking head provides a user-friendly interface, capable to comfort the users as well as attract their attention. This topic has been studied for a decade, and many successful models have been proposed and implemented [11, 41].

The goal of photo-realistic talking head is to make the head

behave like a real person, through rendering a smooth and natural video of articulators in sync with given speech signals. It is a challenging task: rather than just having a static appearance, it has to possess many properties such as convincing plastic deformations of the lips synchronized with the corresponding speech, emotional facial expressions, and realistic head movements. Also, the performance is highly affected by image nuisances, including 3D-pose variation, occlusion and illumination variation, etc.

Such animated talking head can be implemented by selecting an optimal sequence of lip images from a video training dataset, then stitching them back to a background head video. One usual practice is to take a video record of a talking person as the training data with annotated speech. Then through some training algorithm such as Hidden Markov Model (HMM), the audio-visual information is extracted [41]. Thus the visual speech trajectory can be synthesized. Further combining with the technology of Text-to-Speech (TTS), it is possible to generate a head speaking what you want him/her to say by inputting the script of the speech.

Alignment of faces in the video training set is very crucial. Imagine a situation when a human subject being recorded keeps nodding his/her head while speaking, his head pose thus varies among the raw image frames. In this case, if no additional treatment is taken, the synthesized lip motion would definitely be peculiar due to significant misalignment. So face alignment is the first step in generating a talking head. In addition, the alignment has to be very accurate, since a small misalignment may

(a) Before alignment

(b) Aligned with our proposed method

(c) Mouth replaced with other lip shapes

(d) Return to original pose but with new lip shapes

Figure 1.1: Demostration with mouth replacement (illustrated with 20 frames)

already decrease the naturalness of the lips movement perceptually. This motivates our investigation of efficient and precise face alignment algorithms. A demonstration with an interview video obtained from the internet[1] is shown in Figure 1.1. The Poisson image editing [33] is applied in stitching the mouth.

The goal of our thesis is to propose and investigate an efficient face alignment method whose objective is to improve the

[1]The video is obtained from `http://www.beet.tv/2008/09/microsofts-crai.html`.

conventional photo-realistic talking head. For this type of alignment, our mainly concern is about the frontal normal view of the faces. The dataset for training is a video sequence of the same person which may consist of several thousands of image frames. The memory and computational time appear to be a problem for such large datasets. Another difficulty is related to the image nuisances existing in the dataset, which can affect the alignment results. An applicable face alignment algorithm should be able to alleviate the mentioned effects while maintaining the accuracy and efficiency.

1.1.3 Existing methods

Many alignment algorithms which are specialized for faces have been proposed. In the following, an overview of such algorithms will be given.

Statistical Models of Appearance

The most popular alignment methods belong to the class of statistical models of appearance including Active Shape Model (ASM) [10] and Active Appearance Model (AAM) [9]. These methods involve two phases, the training phase and fitting phase. In the training phase, a training set is given; this set is used to generate a parameterized model using the technique of Principal Component Analysis. After we generate the parameterized model, an instance of this model can be fit to a given image so that it can be the best representation of the image in the least squares sense.

AAM can be considered as an enhancement of ASM by adding the gray-scale texture of the faces for the modeling. The optimization strategies are different for ASM and AAM as AAM uses a direct optimization method to match shape and texture simultaneously by considering the global appearance. It was shown that AAM is a reliable algorithm for face alignment [9, 25]. Also, there have been many alternatives and extensions such as Direct Appearance Model (DAM) [27] and progressive AAM [25].

3-D Face Model

The 3-D face model has been applied for face alignment [51, 22, 43]. It involves automatically locating detailed facial landmarks across different viewpoints. Sometimes it can be done by estimating a rigid pose transformation relating a 2-D face image to a 3-D face model. In contrast to the statistical model of appearance, 3-D face model allows a larger range of viewpoints.

Since the datasets we considered are video-based, let us focus on the real-time 3-D model-based pose tracking technique introduced by [43] which is considered to be a fast algorithm for face alignment. Their corresponding method is based on the Bayesian tracking framework. The authors of [43] first define a key-frame, and then obtain a pose posterior distribution which fuses feature correspondence information from both the pervious frame and the key-frame.

Pixel-based Image Alignment

For pixel-based alignment, one basic approach originates from the measures of image similarity [34]. Many congealing algorithms have been proposed such as the unsupervised joint alignment which searches for an alignment that minimizing the sum of entropies of pixel values at each pixel [23], and the least squares congealing procedure suggested in [12]. By stacking the aligned images as the columns of a matrix, the above congealing algorithms require the matrix to be $rank1$. However, in practice, due to some other factors such as large illumination variation in the images, the aligned images may have an unknown rank greater than one. Thus, other rank minimization techniques are proposed in [40] and [32].

Our work can be classified as the pixel-based alignment, which is inspired by the recent development of Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images (RASL) [32]. RASL allows a robust and highly accurate batch alignment of faces in images, despite occlusions, corruptions, and even illumination variations. The idea of iterative linearization for the transformation is explored. This method formulates the batch alignment problem as the solution of convex programs, with the aid of latest advances in rank minimization. Unfortunately, it has limitation on scalability. The memory constraint and computational cost restrain its application on very large datasets, like those we have to deal with in talking head. The detailed of RASL will be discussed in Chapter 5. Although statistical models of appearance and 3-D model provide

fast algorithms, pixel-based alignment allows a more accurate alignment which satisfies our requirement in precision. Thus we choose it to be our foundation.

1.2 Contributions

In this thesis, our concern focuses on aligning the video dataset for photo-realistic talking head which has the properties that the pose changes of consecutive frames are small and the range of face view is confined to a relatively small angle. We use the idea of rank minimization through convex relaxation as well as iterative linearization in RASL. However, RASL fails to deal with very large datasets. We propose two methods to improve the original RASL: multi-RASL and one-by-one approach. We show that our approaches lead to more efficient algorithms while maintaining the same high precision as RASL.

Multi-RASL is a direct extension of the original RASL. Instead of performing batch alignment to all the images in the dataset, we divide the temporal dataset into smaller segments. By choosing a suitable reference image, in theory all the images can be aligned to the reference pose by using RASL on each segment of images. Multi-RASL can tackle the problem due to limitations on memory, since every segment can be computed individually. So, the memory depends on the size chosen for each segment.

In additional to the method of multi-RASL, we further propose an one-by-one approach which applies the latest development of sparse signal representation and sparse corruption tech-

niques using l_1-norm minimization. Instead of aligning the images in batch, we align the images individually using some well aligned images. In other words, if we are given n RASL-aligned images, our approach would try to align the $(n+1)$th image using the information provided. One of the advantages is that it relaxes the constraint of memory; at each time, we only have to store the n RASL-aligned images and the $(n+1)$th image, while for RASL, all the images have to be taken into the memory for the batch alignment.

In our consideration, the input is a set of misaligned faces of the same person obtained from a video sequence and the output is a set of faces which aligned in the same head pose. Obviously, it is useful to align the faces in the frontal normal view. In this case, a reference image or a set of reference images with upright frontal view have to be selected. Original RSAL does not involve any reference images. Therefore, in our proposed methods, we introduce the concept of reference image to our improvement. We choose an upright frontal image as the reference in Multi-RASL while in one-by-one approach we introduce a set of reference images forming a basis.

Moreover, one assumption has been made in our formulation: the face is planar in 3D space. This assumption is reasonable when the angle of face with the frontal normal view is small and the distance between the face and the camera is far relative to the depth of the face. With this assumption, we can apply 2D affine transform to align the misaligned images which makes the problem simpler.

Here we would like to distinguish between our pixel-based

method and homography. In traditional homography, a 2D image of a 3D plane can be transformed to another view easily given at least four pairs of correspondent points. In order to find the pairs, one has to find them either manually or using some automatic feature detection techniques to find some invariant point pairs. Since a face is a deformable object, feature points on the face may vary a lot causing difficulties in getting invariant feature points for doing homography. In our application, while the mouths and eyes are moving, they are not suitable to be the features. Also the nose is not on the face plane, which violates our assumption. The relatively invariant features on the face are the eyebrows. It is possible to extract the positions of the eyebrow corners. Unfortunately, they are nearly collinear which causes ill-conditioned situation especially in low-resolution cases. Therefore, simple homography is not applicable in our task.

By contrast, our method does not require a face feature extraction process. It only needs face detection as preprocessing by which a very rough position of the face is obtained. Besides, all pixels of the face image are involved in the calculation which means that every pixel can be considered as a 'feature point', with the equal importance. Consequently, missing small numbers of these 'feature points' does not affect the results, thus enhancing the robustness of our method.

1.3 Outline of the Thesis

Here the outline of our thesis is briefly described. In Chapter 2, a summary of development of sparse signal representation is presented by first formulating the problems in well-known matrix notations. Then we give a brief introduction to a technique called Basis Pursuit, which replaces l_0-norm minimization by l_1-norm minimization. We will show that Basis Pursuit can solve the l_0-norm minimization when there exists a sufficient sparse solution. Finally, we will mention some of the algorithms for solving the l_1-norm minimization. The work is mainly based on [5], [15] and [45]. The technique of sparse signal representation will be the foundation of our face alignment method and will be the central concept throughout the thesis.

In Chapter 3, we introduce two important developments that are closely related to sparse representation. They are sparse corruptions and Principal Component Pursuit respectively. Here we first discuss the use of l_1-norm minimization to correct the sparse corruptions. Then we link the formulation with that of Least Absolution Deviation (LAD), which has been applied on regression similar to Least Squares. Inspired by the l_1-regularized least squares, we propose the l_1-regularized l_1-error, allowing to search for the sparsest representation and the sparsest error at the same time using an l_1-norm minimization. For the second half of this chapter, the latest sparse and low-rank technique Principal Component Pursuit is reviewed. Finally, we provide here an empirical evaluation of all of our techniques through an experiment on surveillance video.

In chapter 4, the split Bregman algorithm is reviewed for solving the l_1-regularized problem. We start with a brief introduction on the history of its development. Then the formulation of Bregman iteration is given as the basis for split Bregman algorithm. After that we demonstrate the application of split Bregman on our L1LS problem. The work is mainly based on [31, 50, 20].

Chapter 5 starts to consider the problem of face alignment using the perspective of low rank approach. Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images (RASL) is reviewed, based on [32]. Then we propose our improvement on the RASL algorithm, named Multi-RASL, to extend its capability on larger dataset. Their Matlab implementation is discussed and some experiments have been carried out and presented, by considering both small and large datasets.

Finally in Chapter 6 an extension of RASL for video is proposed, using the one-by-one alignment approach. We suggest three choices of l_1-based algorithm which are previously introduced in Chapter 3, including the l_1-regulaized least squares, the l_1-error, and the l_1-regularized l_1-error. Through several experiments we verify the efficacy of our method. It attains comparable quality to RASL while allowing fast and large-scale alignment. The application of our alignment algorithm in photo-realistic talking head is published in a conference paper [47].

Chapter 2

Sparse Signal Representation

2.1 Introduction

In digital signal processing, it is very common to use matrix notations to represent digital signals. For example, a digital gray-scale image can be considered as a matrix with entries representing the intensity, and sometimes it may be convenient to stack it as a vector. A discrete-time signal can also be written as a vector, and it is well-known that such a signal can be transformed to the frequency domain using Discrete Fourier Transform (DFT) which is useful in performing analysis. It involves representing a signal in terms of the Fourier basis, consisting of sine and cosine functions. Actually, DFT not only serves as an effective analysis tool in signal processing, but also be recognized as the key for compression of natural signals and images. One striking example happens in our daily life while we are taking photographs using a digital camera. When we transfer our photos to the computer, usually the computer will automatically help you to transform them to JPEG format. Through the pro-

cess. the photos are compressed by discarding the coefficients corresponding to the high frequencies of Discrete Cosine Transform (DCT), which can be considered as a variant of DFT. This practice works based on the principle that human eye cannot identify the exact strength of brightness of a relatively small area with brightness variation, corresponding to the high frequency components.

In addition to DFT, there are others transformations which also satisfy the compression purpose. Examples include JPEG2000, an improved version over JPEG, that uses wavelet transform instead of DCT. Similar to JPEG, it has been discovered that most natural images have sparse wavelet representations. Thus, it is possible to store an image by using a small number of wavelet coefficients, much fewer than the original image size. This provides many new insights on sparse representations. As we will see later, sparse signal representations have an even more surprising ability other than compression; it can uncover embedded semantic information of the data provided that it exhibits degenerate structure. That means the data can be described originally in a very high-dimension space, but it actually lies on a low-dimensional subspace.

We will use the above property of sparse representations as the main ingredient to deal with our face alignment task. The most useful information for alignment is the face features of the person's face; so as long as the images have a reasonable resolution, it will capture sufficient face features necessary for our purpose. In general, sparse representation technique can exploit the low-dimensional structure (the face features) of the images

without directly extracting it out. However, in our application and in other image processing and computer vision problems, the basis vectors may not be orthogonal to each other like the Fourier or wavelet basis, but is usually over-completed. We build the over-complete dictionary from the sample images which are specifically chosen for our task.

2.2 Problem Formulation

After the introduction from the signal perspective, here we would like to state our problem from the mathematical point of view. Sparse representation can be viewed as solving linear systems of equations. To formulate our problem, let $y \in \mathbb{R}^m$ be the original signal. $D \in \mathbb{R}^{m \times n}$ is the over-complete dictionary, where $m < n$. Our major objective is to solve $y = Dx$ for x. According to the knowledge we get from linear algebra, this is an underdetermine system of linear equations which can have infinite many solutions. To make the problem sensible, we add one requirement: we would like to search for the sparsest x, that is, the solution which contains the least number of nonzero entries. In other words, we would like to decompose y using as few columns of matrix D as possible. This formulation fits our task as it is equivalent to obtaining the sparse representation of a signal.

2.2.1 l_0-norm minimization

Using the common notation of optimization, the above sparse vector searching problem can be written as an l_0-norm mini-

mization problem:

$$\min_x \|x\|_0 \ s.t. \ y = Dx \tag{2.1}$$

Here, the $\|x\|_0$ counts the number of nonzeros in x. Sometimes, it is called an l_0-norm, however it is not a valid norm, since it does not satisfy the positive homogeneity property of norm [1]. We use the l_0-norm to measure the sparsity of a vector. A vector x is known to be K-sparse when $\|x\|_0 = K$. The optimization searches for the sparsest vector x, a vector whose components are the coefficients of the linear combination.

For the underdetermined system of linear equations $Dx = y$, the following questions are of interest:

1. In what situation will the sparest solution be unique?
2. In practice, is it possible to obtain the solution in an efficient way?

In the following section, we will discuss the first question.

2.2.2 Uniqueness

There are two ways to discuss the uniqueness of the sparsest solution of (2.1). The simpler one is to use the spark of the matrix D. Here is the definition of spark.

Definition 2.1. *[15] The **spark** of a given matrix D is the smallest number of columns from D that are linearly dependent.*

Note that spark is not equivalent to the rank of a matrix which is defined as the largest number of linearly independent

[1] The positive homogeneity property $\|ax\|_0 = |a| \|x\|_0$ does not hold for some scalar a.

columns. Comparing to rank, spark is much more difficult to obtain, which requires a combinatorial search over all possible sub-matrices formed using columns of the matrix.

It is simple to illustrate the uniqueness of sparse solution using spark as the criterion:

Theorem 2.1. *[5, 15] If there exists a solution x of $Dx = y$ satisfying $\|x\|_0 < spark(D)/2$, this solution is necessarily the sparsest possible.*

Although the result is simple, the calculation of spark is as difficult as solving (2.1). So it is better to find another criterion for uniqueness. Another way to guarantee uniqueness is to use the coherence of the matrix D, which is defined as follows.

Definition 2.2. *[5, 15] The* ***coherence*** *of a given matrix D is the largest absolute normalized inner product between different columns from D. Let d_k be the kth column of D, the coherence is given by*

$$\mu(D) = \max_{1 \le k,j \le n, k \ne j} \frac{\left| d_k^T d_j \right|}{\|d_k\|_2 \|d_j\|_2} \tag{2.2}$$

Since the inner product of orthogonal columns is zero, orthogonal matrix has zero coherence, sometimes called decoherence. For over-complete dictionary D, it is impossible to have zero coherence. From Theorem (2.1) we see that the optimal solution x of (2.1) is unique when the number of nonzero entries in x is less than half of the number of dependent columns in D. This means that the uniqueness is directly related to the inter-dependence of columns of D. The coherence of a matrix can be viewed as a

weaker measure of the dependence between columns of the matrix. It is found that there is a relationship between spark and coherence, which gives the lower bound of spark as follows.

Lemma 2.1. *[5, 15] The following relationship holds for any matrix D,*

$$spark(D) \geq 1 + \frac{1}{\mu(D)} \tag{2.3}$$

Then with the lower bound of spark, the uniqueness theorem characterizes by the coherence can be written as follows.

Theorem 2.3. *[5, 15] If there exists a solution x of $Dx = y$ satisfying $\|x\|_0 < \frac{1}{2}(1 + 1/\mu(D))$, this solution is necessarily the sparsest possible.*

Therefore, the first question we raised at the end of Section 2.2.1 is answered. The second question about the method to solve the optimal solution of (2.1) is not so straightforward, because of the discontinuity of l_0-lorm. We will introduce in next section a method called Basis Pursuit, which replaces the l_0-norm by l_1-norm.

2.3 Basis Pursuit

Due to the discontinuity of l_0-lorm, the formulation in (2.1) is a NP-hard problem which is not numerically feasible in general. In order to solve (2.1), one way is to regularize the l_0-norm with a continuous or even smooth approximation. There are many possible choices such as replacing the l_0-norm with l_p-norm for some $p \in (0, 1]$ or smooth functions $\sum_i log(1 + \alpha x_i^2)$. It is

found that the choice using l_1-norm is a straight-forward feasible strategy. It is the best convex approximant, and solving of the l_1-norm minimization is possible with many existing optimization algorithms. Researchers sometimes call this method the Basis Pursuit; it which was investigated to a great extent in the past few years, both in empirical and theoretical aspects [14, 15, 21].

2.3.1 From l_0-norm to l_1-norm

By replacing the l_0-norm in (2.1), a new optimization problem is formulated:

$$\min_x \|Wx\|_1 \ \ s.t.\ y = Dx \tag{2.4}$$

Note that the columns in D are not normalized. Since the l_1-norm can be affected by the magnitude of the entries, thus the weighting matrix $W \in \mathbb{R}^{n \times n}$ is needed to compensate the effect. W is a diagonal positive-definite matrix with the diagonal entries $w(i) = \|a_i\|_2$.

To provide a better illustration of the efficiency of using the l_1-norm minimization to solve the sparest solution, let us consider the 2-D case where $m = 1$ and $n = 2$. Figure (2.1) shows the line of $y = Dx$ and the l_1-ball. By changing the size of the l_1-ball, the optimal solution $x^* = [x_1^* \ x_2^*]$ is the point where the l_1-ball touches the line, which is exactly the sparest solution of the system of linear equation $Dx = y$. This simple model can be easily extended to higher dimensions, with the line becoming a hyperplane.

The above demonstration provides the evidence for using l_1-norm minimization to find the sparsest solution. The questions

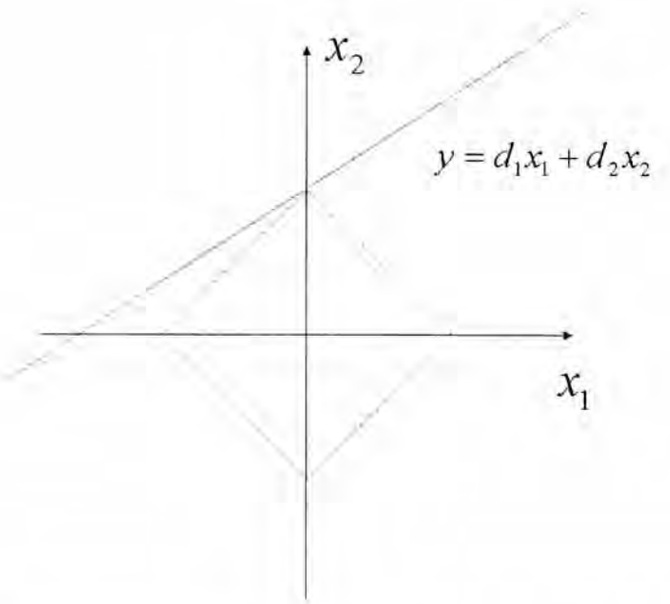


Figure 2.1: Illustration with 2-D case: line of $y = Dx$ and the l_1-ball. The optimal solution $x^* = [x_1^* \; x_2^*]$ is the point where the l_1-ball touches the line.

on when the l_1-norm minimization gives the sparsest solution and how to solve the l_1-norm minimization are of great interest. We will give a brief discussion in the following two sessions.

2.3.2 l_0-l_1 Equivalence

The following theorem shows that Basis Pursuit can solve the sparsest solution when x is sufficiently sparse.

Theorem 2.4. *[5, 15] If there exists a solution x of $Dx = y$ satisfying $\|x\|_0 < \frac{1}{2}(1 + 1/\mu(D))$, this solution is both the unique solution of (2.1) and the unique solution of (2.4).*

Notice that requirement on x in order to ensure unique sparse solution using l_1-norm minimization is consistent with Theorem

2.3. This provides the theoretical support for K-sparse vector x given that the matrix has a coherent $\mu < 1/(2K-1)$. Actually, the theorem gives the sufficient criteria for the l_0-l_1 equivalence, but it is not the necessary one. Empirical evidence [5] shows even for some $\|x\|_0 > \frac{1}{2}(1+1/\mu(D))$ Basis Pursuit still gives the correct sparse vector. That means the above theorem considers the worst-case behavior which is general to all types of matrix. But for some specific ensemble of matrices such as Gaussian matrices, typical behavior is observed; the equivalence between l_0 and l_1 is typical at quite weak level of sparsity [16, 17, 18].

After discussing the criteria for the equivalence of l_1 and l_0-norm minimization, let us introduce some practical methods to solve the l_1-norm minimization problem. One advantage of using BP is that it can be identified as a linear programming problem which has been extensively studied. Many algorithms such as simplex methods, interior-point methods [3], can be applied to solve the problem.

2.4 l_1-Regularized Least Squares

In this section, we will extend the idea that l_1-norm as a representative for sparseness to two more general situations. The first one is the extension of the above formulation to the case with noise. Then for the second one, we consider the problem where the matrix $D \in \mathbb{R}^{m \times n}$ becomes a thin matrix, i.e. $m > n$. We would like to find the sparsest solution x such that the difference between y and Dx is small. Both of the cases leads us to the exploration of l_1-Regularized Least Squares (L1LS).

2.4.1 Noisy case

The underdetermined system of linear equations in (2.4) is useful; however it is not very useful as noise always exists in real world situation. Consider the case where noise is present in original signal y, that is $y = Dx + e$. Here $e \in \mathbb{R}^m$ denotes the noise present. Instead of an equality constraint, we replace it with an inequality constraint of norm of error $\|e\|$ below the noise level ϵ. It is common to use the l_2-norm in this case, such that $\|e\|_2 = \|y - Dx\|_2 \leq \epsilon$. So the problem becomes l_1-norm minimization with inequality constraint, also known in the literature as Basis Pursuit Denoising (BPDN) [8]:

$$\min_x \|Wx\|_1 \ s.t. \ \|y - Dx\|_2 \leq \epsilon \tag{2.5}$$

where W is again the diagonal positive-definite weighting matrix. With an appropriate Lagrange multiplier λ, the following unconstrained optimization problem gives the same solution as in 2.5:

$$\min_x \frac{1}{2} \|y - Dx\|_2^2 + \lambda \|Wx\|_1 \tag{2.6}$$

Now it becomes an L1LS problem.

2.4.2 Over-determined systems of linear equations

Now let us consider the case where $m > n$, then $Dx = y$ becomes an over-determined system of linear equations. One possible question is, can we find a very sparse solution x such that the "difference" between y and Dx is small? One common approach is to represent the "difference" in terms of l_2-norm $\|y - Dx\|_2$.[2]

[2]One reason for choosing the least squares is that if the noise is in normal distribution (Gaussian noise), least squares corresponds to the maximum likelihood criterion

The objective function would be very similar to (2.6):

$$\min_x \frac{1}{2} \|y - Dx\|_2^2 + \mu \|Wx\|_1 , \tag{2.7}$$

where μ is the weight that trades off the least square error and the sparsity of x.

The above formulation exploits the property that the l_1-norm can capture sparsity. The regularization improves over ordinary Least Squares (LS) in the situation where the signal y contains noises other than Gaussian one. LS representation would use all the atoms to compensate the non-Gaussian noise, sacrificing the important features of the signal. The l_1-regularization chooses the most representative atoms from the dictionary D to express y to reduce the distortion by outliers and thus enhances the robustness of LS.

To solve L1LS, there are many possible methods, including the general convex optimization techniques such as interior-point methods [3]. Although these algorithms give high-accuracy solution, it is known that in general, the computational cost is too expensive. It is not practical in many of the applications requiring both fast and accurate solution.

There are many fast l_1-norm minimization algorithms introduced to seek an alternative to solve (2.7) approximately, including the gradient projection methods [19], homotopy methods [29], iterative shrinkage-thresholding methods [13], proximal gradient methods [30], and augmented Lagrange multiplier methods [49]. A review paper [48] gives a comparison of the above listed algorithms.

In this thesis, we investigate a relatively new algorithm called

split Bregman method for solving our l_1-norm minimization problem. It is shown that Bregman method [20] gives both accurate and fast results for our applications. In [50], it mentions that the split Bregman method is equivalent to the iterative process of augmented Lagrange multiplier method under some conditions. The detail descriptions and derivations will be given in the later chapter.

2.5 Summary

In this chapter, we have first presented the formulation of sparse signal representations using the l_0-norm minimization and discussed the condition of uniqueness. Then the Basis Pursuit approach is suggested to replace the l_0-norm by l_1-norm. The l_0-l_1 equivalence also enables the L1LS which apply in noisy case as well as over-determined systems of linear equations. In order to solve L1LS and other l_1-problems, the split Bregman algorithm will be introduced in the succeeding chapter.

Chapter 3

Sparse Corruptions and Principal Component Pursuit

3.1 Introduction

In image processing, in addition to the signal noise, there can be many different kinds of variations that can lower the information quality, such as occlusions and changes in illumination conditions, etc. Usually the occlusions would not cover all the features; otherwise the useful information can no longer be extracted. Therefore it is reasonable to consider partial occlusions, or, generally, sparse corruptions. We have two major objectives. First we have to seek a way to correct the distortions. It is found that this aim can be achieved by using the l_1-norm minimization. Besides the correction, we are also interested in the content of images. In other words, we want to extract the information inside the images. Robust Principal Component(RPCA) is proposed for this purpose, provided that the hidden content has low dimensional hidden structure. It goes through a rank minimization, which can be viewed as a variant of l_0-norm minimization

since minimizing the rank of a matrix is equivalent to minimizing the number of nonzero entries in the eigenvalues of the matrix. In principle, RPCA is able to extract the content and at the same time correcting the sparse corruptions by applying the technique of Principal Component Pursuit. In the following, both sparse corruptions and RPCA are explained through detailed formulations as well as experiments.

3.2 Sparse Corruptions

In this section, we first define sparse corruptions and formulate the correction problem for the corruptions using the l_1-norm minimization. Then we mention the connection between our l_1-error formulation with Least Absolute Deviations (LAD). Then we end this section by proposing the formulation of l_1-regularized l_1 error.

3.2.1 Sparse Corruptions and l_1-Error

Sparse corruptions refers to the corruptions appearing in a small number of pixels. The locations and magnitudes of the corruptions can be arbitrary. That means we do not assume any patterns or statistical models for the corruptions. Recall that in Chapter 2, we wanted to find a representation for a signal or an image using a given dictionary. Previously we have mentioned how to use l_1-norm minimization to search for the sparsest representation. Here we are not going to constrain on the representation coefficients. Instead, we would like to obtain a

representation such that its difference from the given signal or image vector has the smallest number of nonzero entries. Same as in Chapter 2, let $y \in \mathbb{R}^m$ be the given signal and $D \in \mathbb{R}^{m \times n}$ be the dictionary. We can formulate our problem as:

$$\min_x \|y - Dx\|_0 , \tag{3.1}$$

where $\|\cdot\|_0$ represents the l_0-norm which counts the number of nonzero entries.

Sometimes, (3.1) can be rewritten in an equivalent form:

$$\min_x \|e\|_0 \ \ s.t.\ y = Dx + e \tag{3.2}$$

where e is known as the corruptions or the error term.

In fact, (3.2) is a l_0-norm minimization similar to (2.1). So applying the same argument as in Chapter 2, we replace the l_0-norm by l_1-norm, and we get

$$\min_x \|e\|_1 \ \ s.t.\ y = Dx + e \tag{3.3}$$

(3.3) is an l_1-norm minimization for the error term. For convenient, we call this l_1-error approach. It is easy to see that (3.3) can be recast in the form of (2.4):

$$\min_w \|\Phi w\|_1 \ \ s.t.\ y = \Psi w \tag{3.4}$$

where $w = [x\ e]^T$, $\Phi = [\mathbf{0}_{n \times n}\ \mathbf{I}_{n \times m}]$ and $\Psi = [D\ \mathbf{I}_{m \times m}]$. Therefore the theorems in Section 2.3.2 are valid for our new optimization problem. We will see how to implement a fast algorithm in Chapter 4.

3.2.2 l_1-Error and Least Absolute Deviations

Our l_1-error approach is not a novel one. In fact, it has been studied for decades in the field of regression analysis. In this section, we provide a brief review on how this can be applied in regression, and also make a connection between regression and our problem for a better insight.

In regression analysis, the goal is to fit a set of data using a function. One usual attempt is to use least squares. Assumeing the errors are identically and independently distributed, the least squares is proven to be the most efficient among the unbiased estimation methods. However, for some cases where many outliers exist, due to the quadratic weighting of least squares, it becomes highly unstable and sample dependent. Thus, it raises the research of robust estimation.

One candidate for robust estimation is Least Absolute Deviation (LAD) [35, 1, 2, 37]. Consider the regression model with sample size n:

$$y_i = x_i^T \beta + \epsilon_i \tag{3.5}$$

where y_i is the dependent variable for $i = 1, \ldots, n$, $x_i = [1, x_1, \ldots, x_k]^T$ is a $(k+1)$-vector of explanatory variables, $\beta_i = [1, \beta_1, \ldots, \beta_k]^T$ is the regression coefficient vector and ϵ_i are uncorrelated disturbance terms. Then the LAD estimator of the parameter vector β in (3.5) is the one that minimizes

$$\sum_{i=1}^{n} |y_i - x_i^T \beta| \tag{3.6}$$

Notice that it differs from least squares only by minimizing the sum of absolute values instead. Although most papers do not

mention about the l_1-norm, but it is obvious that (3.6) is exactly the same as our least l_1-error problem (3.3) where β corresponds to x in our problem.

Comparing LAD with the least squares, it can be found that LAD is robust to outliers. Consider a signal with several entries corrupted significantly; the least square solution would smooth out the outliers by compensating with the other entries, causing an overall change in resulting linear combination which differs a lot from what we would expect. However, LAD can pick out the outliers, keeping the others unaffected, so the combination can preserve the features of the uncorrupted signal. This explains why we can correct the sparse corruption using the least l_1-error approach.

Another important property of the regression perspective is that the LAD estimator β equals the maximum likelihood estimator if the disturbances follow the Laplace distribution with parameter $\lambda > 0$:

$$f(\epsilon_i) = \frac{1}{2\lambda} exp(-\frac{|\epsilon_i|}{\lambda}) \tag{3.7}$$

In comparison, the least squares estimator is the same as the maximum likelihood estimator for Gaussian disturbances. This provides a new insight to the least l_1-error problem.

3.2.3 l_1-Regularized l_1-Error

In Section 2.4, we extend the ordinary least squares by using an l_1 regularization to constrain the sparsity of representation and formulate the l_1-regularized least squares (L1LS). Here we

utilize the same idea to the l_1-error problem. We propose the l_1-regularized l_1-error (L1L1) for the problem that sparsity in both representation and disturbance vector are of concern.

The L1L1 can be expressed as the following optimization problem:

$$\min_x \|y - Dx\|_1 + \mu \|Wx\|_1 \tag{3.8}$$

or

$$\min_{x,e} \|e\|_1 + \mu \|Wx\|_1 \ \ s.t.\ e = y - Dx \tag{3.9}$$

Here μ is the weight which trades off the least l_1 difference and the sparsity of x, while W is the weighing matrix that compensates the effect of imbalance within columns of the dictionary D as they may not be normalized.

Notice that (3.9) can be transformed to the general l_1-norm minimization problem:

$$\min_w \|\Phi w\|_1 \ \ s.t.\ y = \Psi w \tag{3.10}$$

where $w = [x\ e]^T$, $\Phi = [\mu W\ \mathbf{I}_{n\times m}]$ and $\Psi = [D\ \mathbf{I}_{m\times m}]]$. Thus the theorems in Section 2.3.2 is valid for our new optimization problem.

Given that the dictionary D consists of training data from separable groups, for example, D is a face dataset of several individuals and there are a few face photos for each person in face recognition. It is reported in [46] that one version of L1L1 is applicable in face recognition under occlusions. We will exploit this classification property of L1L1 to a practical example in Section 3.4 where we deal with three combined surveillance videos. Besides, we will show that our proposed L1L1 approach

gives an efficient and accurate result in the application of face alignment.

3.3 Robust Principal Component Analysis (RPCA) and Principal Component Pursuit

Consider the case where we have a set of image data sparsely corrupted and we know that the data all lie near some low-dimensional subspace. We would like to eliminate the corruptions and retain the useful low-dimensional information. [6] introduces a technique of rank minimization that can exactly fulfill our task, which is named Robust Principal Component Analysis (RPCA). It is developed recently through the advancement of transforming intractable sparse problem to a feasible convex optimization problem. In this section, we will give an introduction to RPCA by distinguishing it from classical Principal Component Analysis (PCA) and further introduce the Principal Component Pursuit, a tractable convex optimization which allows solving the RPCA practically.

3.3.1 Principal Component Analysis (PCA) and RPCA

PCA is a mathematical procedure which converts a set of observations of possibly correlated variables into a set of values of uncorrelated variables called principal components. This conversion is defined such that the first principal component has as large variance as possible and each succeeding component in turn has the highest variance possible under the constraint that

it is orthogonal to the preceding components.

PCA is particularly useful in image classification because digital images nowadays usually consist of millions of pixels, however no matter how high the resolution is, the content remains the same. That means such data have low dimensionality. PCA is efficient for the linear case where the data all lie near some low-dimensional subspace. If we stack all the images as column vectors of a matrix M, the matrix would be low-rank or approximately low-rank.

Basically, PCA is an orthogonal transformation process which can be achieved by singular value decomposition (SVD). When PCA is applied for dimensionality reduction, it can be formulated as a low-rank matrix approximation problem for matrix $M \in \mathbb{R}^{m \times n}$:

$$L_0 = arg \min_L \|M - L\|_F \ s.t. \ rank(L) \leq k \tag{3.11}$$

where $L_0 \in \mathbb{R}^{m \times n}$ is the rank-k estimate of M and $\|\cdot\|_F$ is the Frobenius norm, which is the sum of squares of all entries.

Although PCA is the most widely used statistical tool for data analysis and dimensionality reduction, unfortunately it is brittle with respect to grossly corrupted observation: a single grossly corrupted entry in D could cause the solution of (3.11) deviates a lot from the true L_0. RPCA improves the classical PCA so that it is able to correct the large value corruptions while finding the best rank-k estimate of L_0, given that the corruptions are sparse, which means the number of corrupted entries in M is sufficiently small.

3.3.2 Principal Component Pursuit

Here is the formulation of RPCA with reference to [6]. Given a large data matrix $M \in \mathbb{R}^{m \times n}$, we would like to decompose it as

$$M = L + S \tag{3.12}$$

where $L \in \mathbb{R}^{m \times n}$ is a low-rank matrix and $S \in \mathbb{R}^{m \times n}$ is sparse. The exact rank of L and the locations of the nonzero entries of S is not given. In order to search for the sparse and low-rank solution, the problem can be written as the following optimization:

$$\min_{L,S} rank(L) + \gamma \|S\|_0 \ \ s.t.\ M = L + S \tag{3.13}$$

Here $\gamma > 0$ is a parameter that trades off the rank of L against the sparsity of S. In fact, (3.13) is the formulation of RPCA mentioned in Section 3.3.1. However, since the rank and l_0-norm are discontinuous and even non-convex. The computation is impossible for large-scale problem as it is NP-hard. Fortunately, the recent development of sparse representation which we have introduced in Chapter 2 provides an inspiration. It shows the possibility to approximate a discontinuous problem to a convex one through the convex relaxation technique. Also the performance is guaranteed. The convex relaxation of (3.13) can be achieved by replacing the rank with the nuclear norm and the l_0-norm by l_1-norm. The nuclear norm of L is defined as $\|L\|_* = \sum_{i=1}^{m} \sigma_i(L)$ which is the sum of all singular values of L. It is easy to notice that the idea is the same as the l_0 to l_1 relaxation, since $rank(L)$ equals to the number of nonzero singular values of L. After applying this relaxation, (3.13) yields a new

optimization problem which is called the Principal Component Pursuit (PCP):

$$\min_{L,S} \|L\|_* + \lambda \|S\|_1 \ s.t.\ M = L + S \tag{3.14}$$

Here λ is a parameter. Surprisingly, [6] shows that λ can be a fixed constant for PCP to perfectly decompose into the low-rank and the sparse component, provided that the rank of the low-rank matrix L is not too large, and the sparse matrix S is reasonably sparse. [6] suggests that $\lambda = 1/\sqrt{\max(m,n)}$. PCP gives a new objective function which is non-smooth, but at least continuous and convex.

There are several existing algorithms for solving matrix completion which is also applicable in solving PCP, such as the iterative thresholding [44], the accelerated proximal gradient [39] and the augmented Lagrange multiplier (ALM) [28]. In our experiment, we choose the inexact ALM approach, and implement according to the algorithm described in [28].

3.4 Experiments of Sparse and Low-rank Approach on Surveillance Video

Figure 3.1 shows all the frames of the combined videos. There are altogether 210 frames of dimension 192×144. The three videos contain 100, 10 and 100 frames respectively. Each video sequence is taken from a stationary camera and hence the background is unchanged. In the video there are some moving objects such as cars or humans.

The simulation is run in Matlab. The frames are stacked as the columns of the input matrix M. We set the weight to be $1/\sqrt{m}$ where $m = 192 \times 144$ which is the size of each frame. The resulting low-rank matrix L has $rank = 109$. The decomposition output L and S is displayed in Figure 3.2 and 3.2.

Figure 3.2 shows that RPCA performs well in the task of background modeling. It has the ability to extract the moving objects (foreground) which only occupy relatively small areas in the images. The background information is obtained in the low-rank matrix L, while the moving objects, acting as occlusion to the background, are extracted in sparse matrix S.

3.4.1 Least Squares

First, before discussing the features of l_1-techniques, we use the ordinary least squares to obtain a contrast. Figure 3.4 shows the results using least squares. Figure 3.4a show the original frame on the top and the linear combination by D on the bottom, while in Figure 3.4b the entries of vector x is displayed in graph. It represents the coefficient for the combination of the columns of D in representing the test frame. It is found that least squares tried to use all the columns in D to form the linear combination of the test frame.

3.4.2 l_1-Regularized Least Squares

Next, we applied to L1LS. The weight μ is set to be 10^{-3}. The result is shown in Figure 3.5. In contrast to least squares case, L1LS gives a vector x which allows us to identify which video the

frame is corresponding to. As seen in Figure 3.5b, the highest peak appears at the 110^{th} entry. Therefore, L1LS is better than least squares in our application.

3.4.3 l_1-Error

We know that the foreground objects are sparse, i.e. they only cover relatively small areas on the images. Therefore, we try to apply the l_1-error algorithm. Figure 3.6 gives the output of l_1-error. As seen in the bottom of Figure 3.6a, the combination can hardly represent the selected frame. It is not surprising as we restrict the difference between the selected frame and its representation using D to be sparse, without giving any constraints on the combination.

3.4.4 l_1-Regularized l_1-Error

To achieve our goal of getting the best representation for the chosen frame, we use the L1L1 algorithm. The output is shown in Figure 3.7 which satisfied all our needs. We can see a clear broad peak in Figure 3.7b, showing that most ingredient in the combination is corresponding to the 100^{th} to 110^{th} frames of the combined video. That range of frames is exactly the video in which the selected frame belongs to. As a result, we confirm that L1L1 is the best in decomposing the foreground and background in a single image using the low-rank and sparse approach. This special feature of L1L1 inspires us to use this on the application of face alignment, which will be introduced in Chapter 6.

3.5 Summary

A series of algorithms using l_1-norm minimization to compensate sparse corruptions or noise is introduced. It includes the l_1-error and l_1-regularized l_1-error. Besides, the relationship between l_1-error and least absolute deviation is discussed. The idea of replacing counting number of nonzero entries in a vector by computing the absolute sum of all entries gives rise to Principal Component Pursuit (PCP). Using PCP, robust Principal Component Analysis can be done. In this chapter, we provide an experiment using sparse and low-rank approach to separate the foreground and background in surveillance videos. It shows the ability of PCP to model the background, as well as the advantage of l_1-regularized minimization in finding the best representation given the dictionary of more than one surveillance videos. This discovery enables us to use these properties of sparse and low-rank algorithms to our face alignment. In addition, it raises the possibility to apply similar methods in video segmentation and other video-based applications.

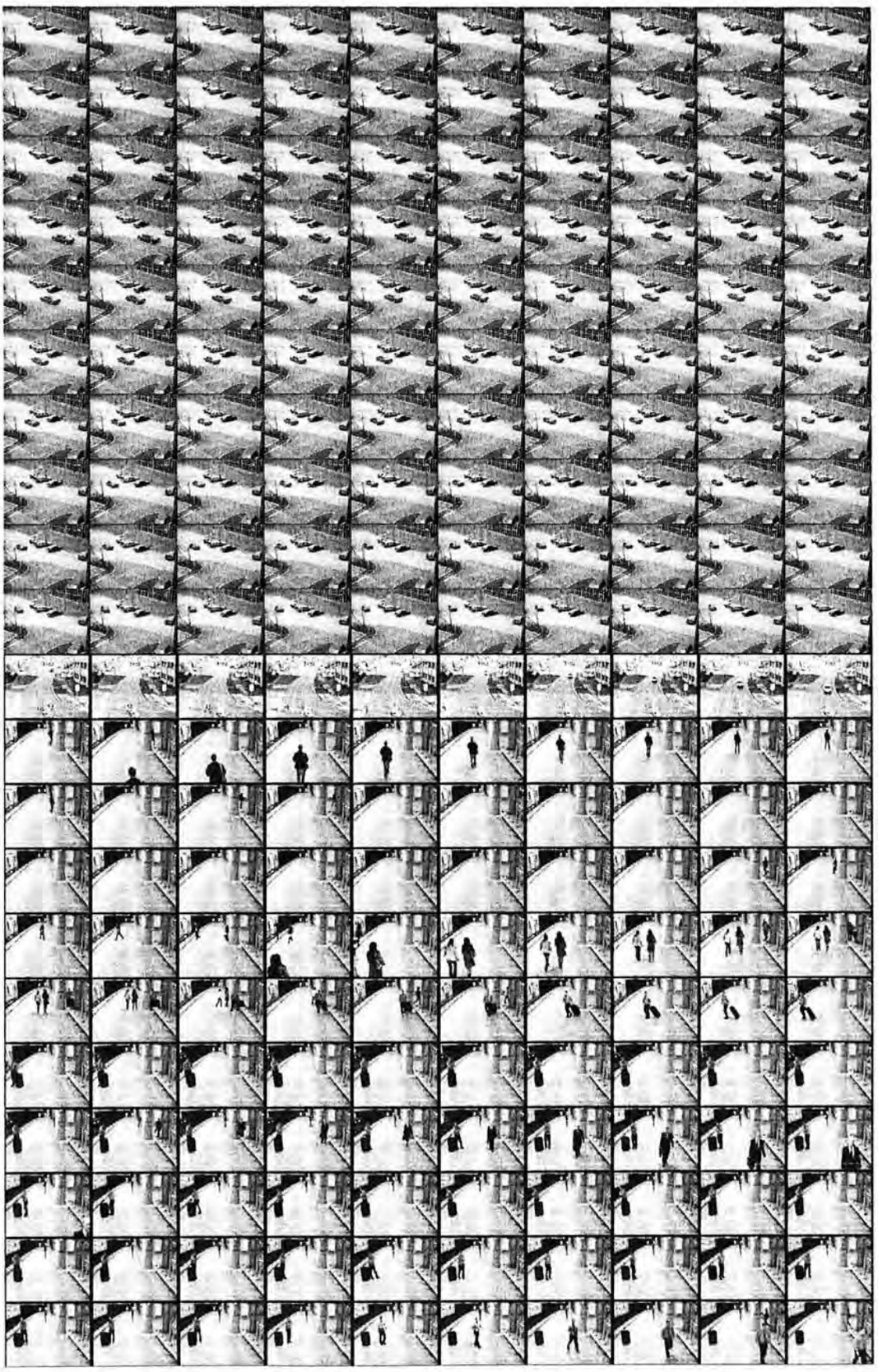


Figure 3.1: The original 210 frames video combined by three independent surveillance

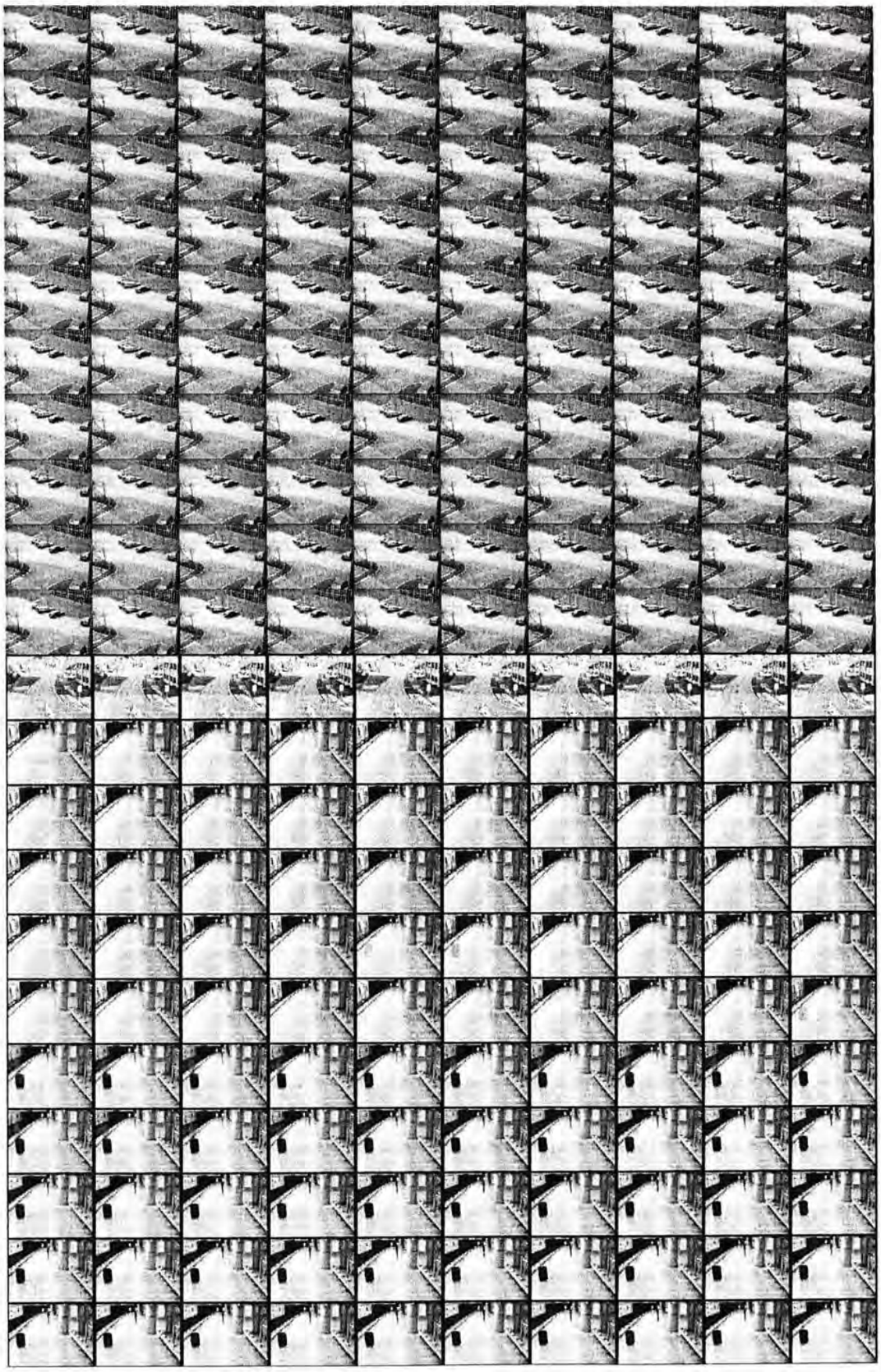


Figure 3.2: Low-rank L

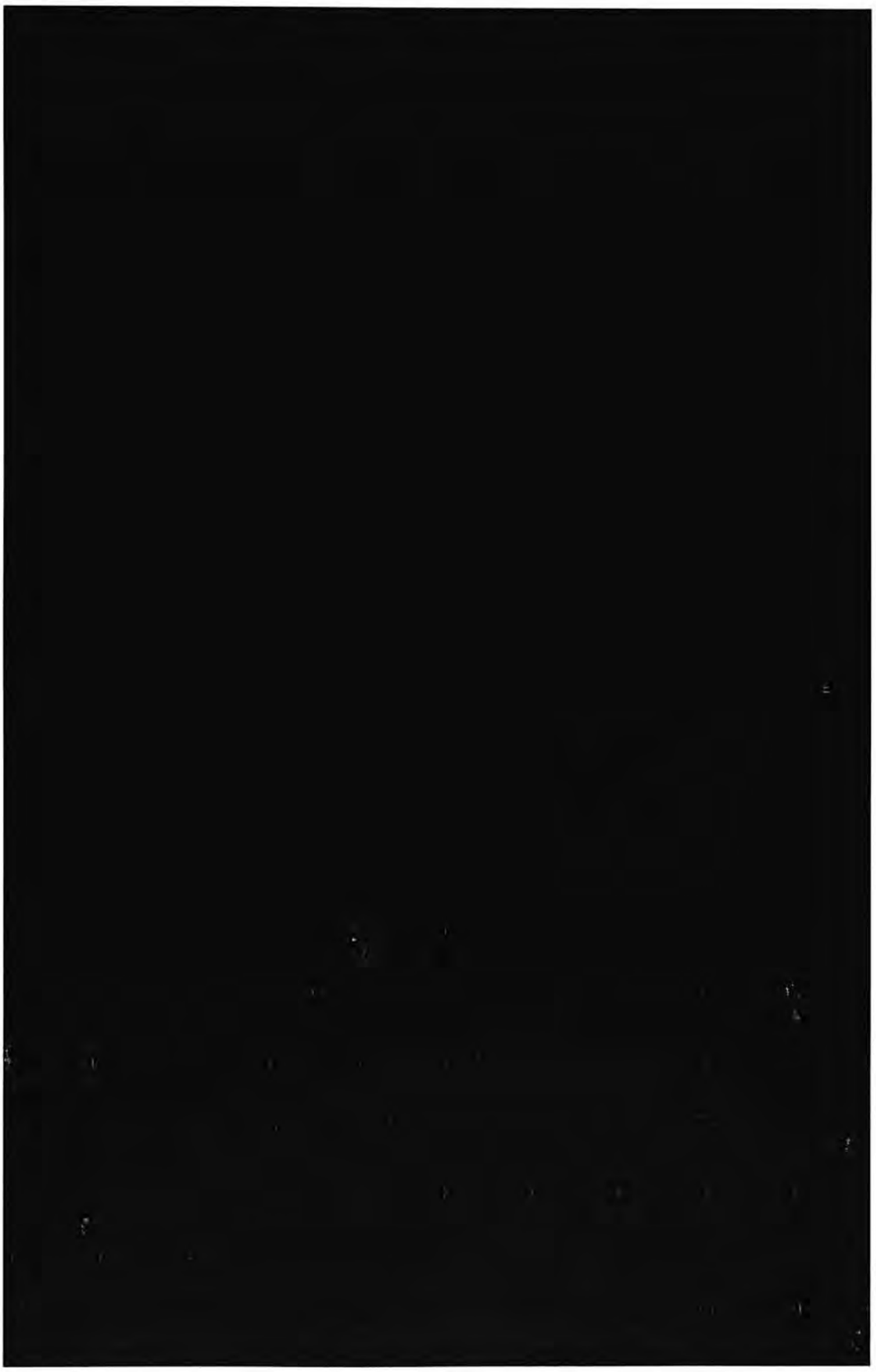
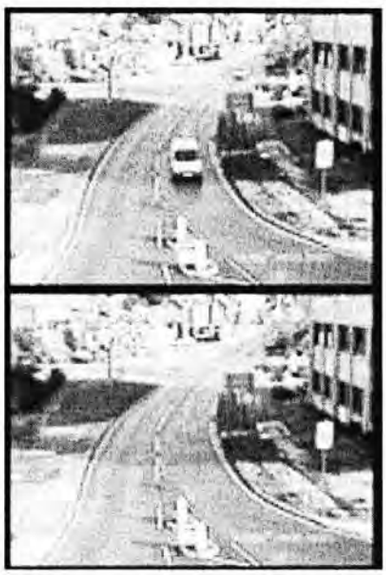
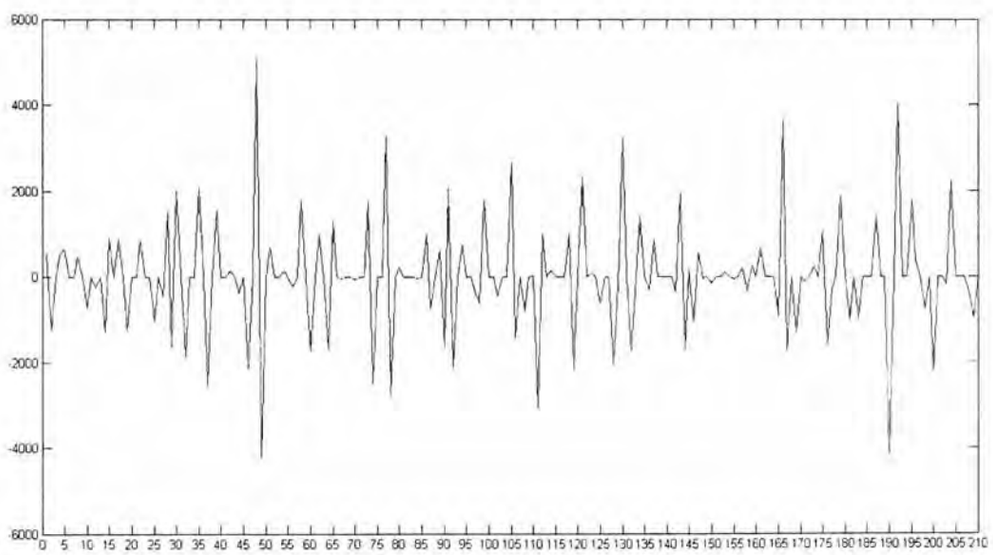


Figure 3.3: Sparse E

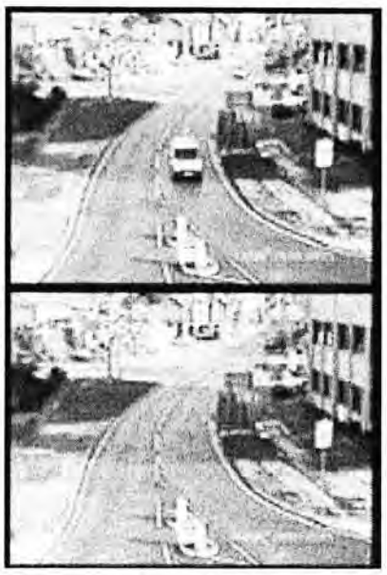


(a) The 110^{th} frame (top) and linear combination using D (bottom)

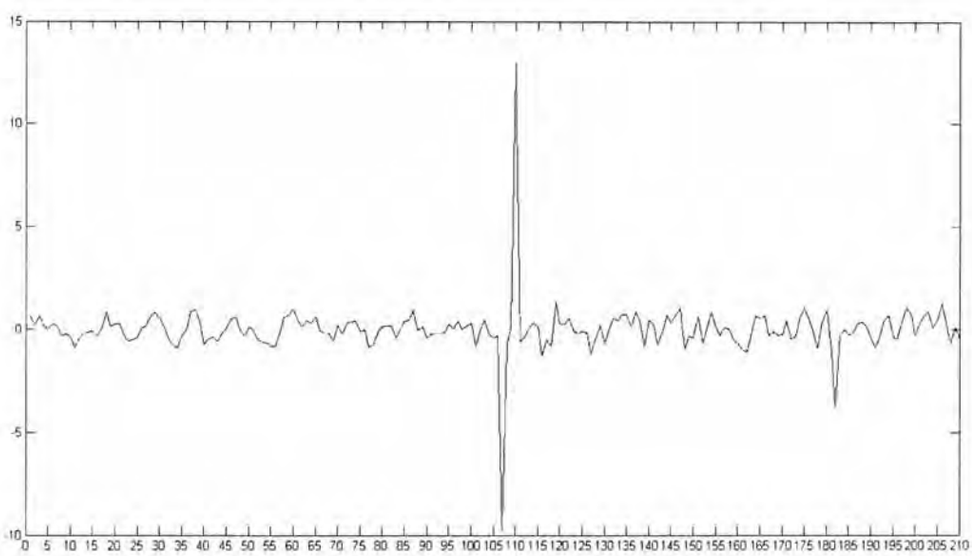


(b) Entries in x

Figure 3.4: Decomposition using least squares



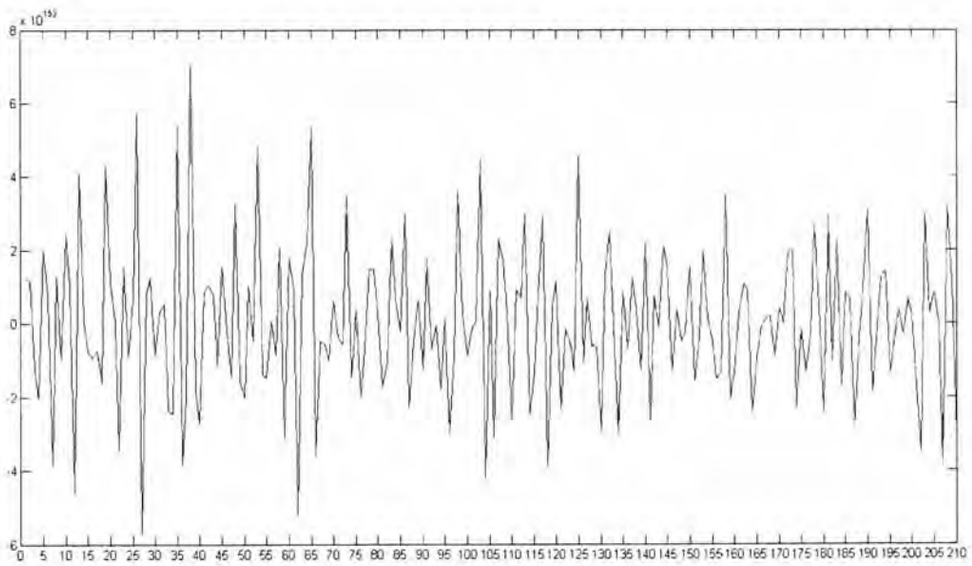
(a) The 110^{th} frame (top) and linear combination using D (bottom)



(b) Entries in x

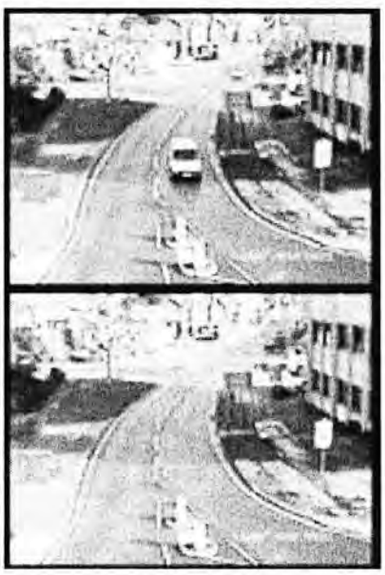
Figure 3.5: Decomposition using l_1-regularized least squares

(a) The 110^{th} frame (top) and linear combination using D (bottom)

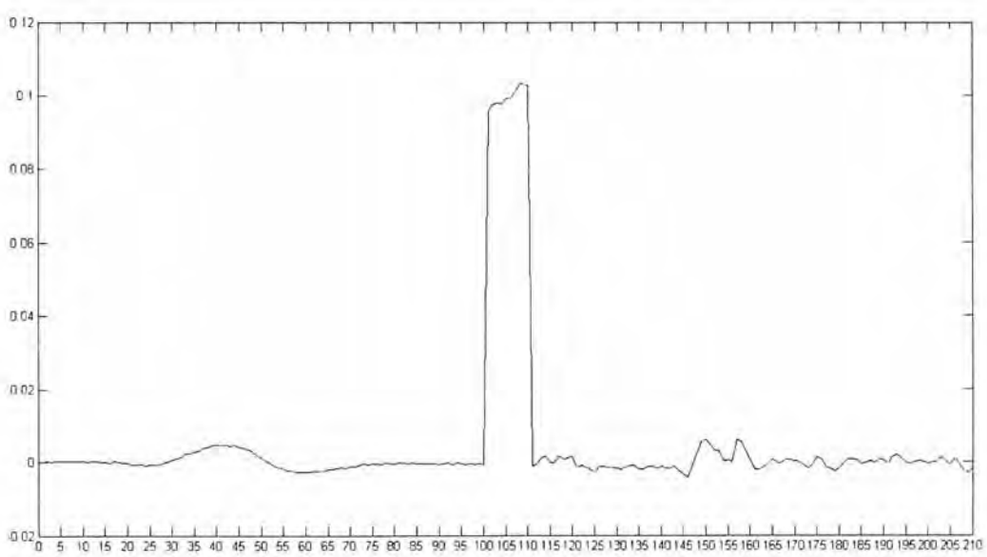


(b) Entries in x

Figure 3.6: Decomposition using l_1-error



(a) The 110^{th} frame (top) and linear combination using D (bottom)



(b) Entries in x

Figure 3.7: Decomposition using l_1-regularized l_1-error

Chapter 4

Split Bregman Algorithm for l_1-Problem

4.1 Introduction

In order to solve the L1LS problem in (2.6) in chapter 2, many algorithms have been proposed such as the interior-point methods [3]. The problem is reformulated as a quadratic programming problem in [26]. There are other toolbox which applies interior-point methods including the famous solver for l_1 problem for compressed sensing called l_1-magic [7], which considers the l_1-norm minimization in compressed sensing problem as a second order cone program, and employs the logarithmic barrier potential for inequality constraints. Although the interior-point approach allows tractable algorithm giving reliable solution, the computation is generally slow and unable to deal with large-scale problems. In many applications in signal and image processing, a fast algorithm is required. Efforts have been made in the past few years to develop trustworthy algorithms for tackling the l_1 problems. One of the candidates is the split Bregman method.

The Bregman method was first introduced in [31] and applied to the Rudin-Osher-Fatemi (ROF) image denoising model with non-smooth total variation regularization:

$$u = arg\min_u \mu \int |\nabla u| + \frac{1}{2}\|u - f\|_2^2 \tag{4.1}$$

where u is the denoised image, f is the observed noisy image and μ is a positive parameter related to signal-to-noise ratio.

In [20], the similarity of ROF denoising and Basis Pursuit is discovered and exploited to suggest new efficient l_1 solver using the extension of ordinary Bregman method: split Bregman method. In general, split Bregman can solve the problem

$$\min_u \|\phi(u)\|_1 + E(u) \tag{4.2}$$

under the assumption that E is convex and ϕ is convex and differentiable.

Through introducing the Bregman distance [4], the split Bregman iterations can solve the l_1 problem by repeating simple operations such as shrinkages, matrix multiplications, and a small number of matrix inversions.

4.2 Bregman Distance

Originally, Bregman iteration originated in functional analysis for solving convex optimization problems [4]. To introduce Bregman algorithm, we begin with the concept of Bregman distance.

Definition 4.1. *[4] The Bregman distance of J between u and v is*

$$D_J^p(u, v) = J(u) - J(v) - \langle p, u - v\rangle \tag{4.3}$$

where $p \in \partial J(v)$.

$\partial J(v)$ is the subgradient of J at v which is defined by $\partial J(v) = \{p : J(u) \geq J(v) + \langle p, u - v \rangle, \forall u\}$.

Obviously, the Bregman distance is not the usual distance as it is not symmetric in general. However, it does measure the closeness of any two points u and v because $D_J^p(u, v) \geq 0$ since J is convex and $D_J^p(u, v) \geq D_J^p(w, v)$ for any w on the line segment joining u and v. Figure 4.1 uses a smooth convex function on 2-D plane to illustrate the Bregman distance. For the smooth convex function, the subgradient p of J at point v is the ordinary gradient, i.e. the slope. The straight line that touches $J(v)$ can be viewed as the first-order expansion of J at v. The Bregman distance $D_J^p(u, v)$ is the difference between the points $J(u)$ and $J(v) + p^T(u - v)$. It is easy to see that $D_J^p(u, v) \neq D_J^{p^u}(v, u)$ in our example. For the non-smooth case, the non-differentiable point has more than one subgradients. However by picking a fixed subgradient p, we get $D_J^p(u, v) \geq D_J^p(w, v)$ for all w in between u and v.

4.3 Bregman Iteration for Constrained Optimization

First let us consider the constrained optimization problem:

$$\min_u J(u) \ s.t. \ Au = b \tag{4.4}$$

where J is a convex function on u, and $A \in \mathbb{R}^{m \times n}$. In usual approach, (4.4) will be an approximation to an unconstrained

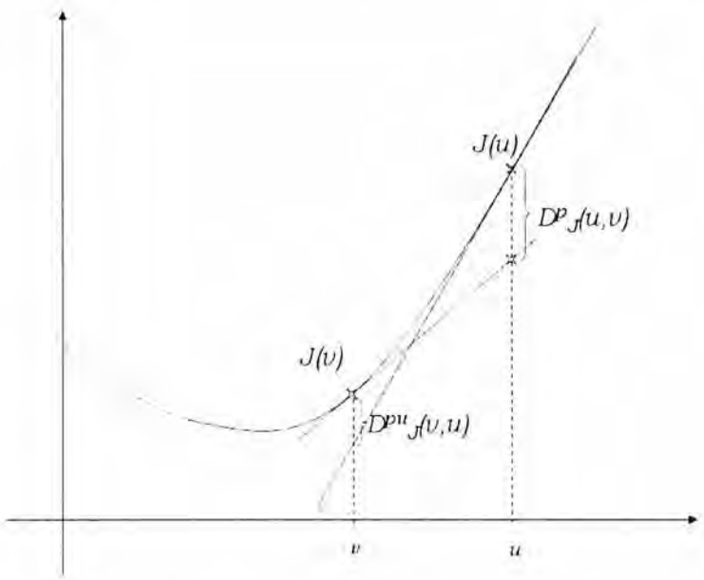


Figure 4.1: Illustration of Bregman distance.

optimization problem with a penalty function:

$$u_{k+1} = arg\min_u J(u) + \frac{\lambda_k}{2} \|Au - b\|_2^2 \tag{4.5}$$

where $\lambda_1 < \cdots < \lambda_N$ is an increasing sequence of weights for the penalty function. For better approximation of $Au - b \approx 0$, λ_N should be chosen as large as possible. Unfortunately, for many algorithms such as the Newton-type method, ill-conditioning occurs for very large λ_k as these methods depend on the eigenvalues structure o the Hessian of the objective function.

To solve the constrained optimization problem (4.4), Bregman iteration gives an efficient and reliable approach, which prevents the above ill-conditioning problem, by recasting (4.4) into an unconstrained problem using a quadratic penalty function:

$$\min_u J(u) + \frac{\lambda}{2} \|Au - b\|_2^2 \tag{4.6}$$

Instead of letting $\lambda \to \infty$, we iteratively minimize:

$$\begin{aligned} u^{k+1} &= \min_u D_J^{p^k}(u, u^k) + \frac{\lambda}{2} \|Au - b\|_2^2 \\ &= \min_u J(u) - \langle p^k, u - u^k \rangle + \frac{\lambda}{2} \|Au - b\|_2^2 \end{aligned} \tag{4.7}$$

for $k = 0, 1, \ldots$ with $u^0 = 0$ and $p^0 = 0$. Previously we only assume J is convex, but not necessarily differentiable. Thus, the subgradient of J may be unique. But we can get the update rule for p^{k+1} directly as 0 is included in subgradient of $J(u) - \langle p^k, u - u^k \rangle + \frac{\lambda}{2} \|Au - b\|_2^2$, i.e. $0 \in \partial J(u^{k+1}) - p^k + \lambda A^T(Au^{k+1} - b)$. Therefore,

$$p^{k+1} = p^k - \lambda A^T(Au^{k+1} - b) \tag{4.8}$$

[20] claims a simplified algorithm that is equivalent to (4.7) and (4.8):

$$u^{k+1} = \min_u J(u) + \frac{\lambda}{2} \left\|Au - b^k\right\|_2^2 \tag{4.9}$$

$$b^{k+1} = b^k + b - Au^k \tag{4.10}$$

However, they did not give the proof. Here we provide a simple derivation. By (4.8), as $p^0 = 0$, we have

$$p^1 = -\lambda A^T(Au^1 - b^1) \tag{4.11}$$

where b^1 is defined as b for convenient,

$$\begin{aligned} p^2 &= -\lambda A^T(Au^1 - b^1) - \lambda A^T(Au^2 - b) \\ &= -\lambda A^T(Au^2 - (b^1 + b - Au^1)) \end{aligned} \tag{4.12}$$

Then we let $b^2 = b^1 + b - Au^1$, (4.12) becomes

$$p^2 = -\lambda A^T(Au^2 - b^2) \tag{4.13}$$

By simple induction, it is obvious that we can get

$$p^k = -\lambda A^T(Au^k - b^k) \tag{4.14}$$

with a new sequence b^{k+1} defined as

$$b^{k+1} = b^k + b - Au^k \tag{4.15}$$

Now consider (4.7), as u^k is a constant in u, (4.7) becomes

$$u^{k+1} = \min_u J(u) - \langle p^k, u\rangle + \frac{\lambda}{2}\|Au - b\|_2^2 \tag{4.16}$$

Using (4.14) and (4.15), (4.16) can be further simplified:

$$\begin{aligned}
u^{k+1} &= \min_u J(u) - \lambda\langle A^T(Au^k - b^k), u\rangle + \frac{\lambda}{2}\|Au - b\|_2^2 \\
&= \min_u J(u) + \lambda(Au^k - b^k)^T au + \frac{\lambda}{2}(Au - b)^T(Au - b) \\
&= \min_u J(u) + \frac{\lambda}{2}\Big(\|Au\|_2^2 - 2\big(b - (Au^k - b^k)\big)^T Au + \|b\|_2^2\Big) \\
&= \min_u J(u) + \frac{\lambda}{2}\big\|Au - \big(b - (Au^k - b^k)\big)\big\|_2^2 \\
&= \min_u J(u) + \frac{\lambda}{2}\big\|Au - b^k\big\|_2^2
\end{aligned} \tag{4.17}$$

Hence (4.9) and (4.10) are derived.

4.4 Split Bregman Iteration for l_1-Regularized Problem

After introducing the Bregman iteration for constrained optimization, in the following part we will show how to solve our l_1 problem using the improved version of Bregman iteration: the split Bregman.

4.4.1 Formulation

Suppose we are given the optimization problem with an l_1-term:

$$\min_u \|\phi(u)\|_1 + E(u) \tag{4.18}$$

We rewrite the unconstrained optimization problem (4.18) to an constrained one by operator splitting

$$\min_{u,d} \|d\|_1 + E(u) \; s.t. \; d = \phi(u) \tag{4.19}$$

Then according to Bregman formulation (4.7) and (4.8), (4.19) can be converted to a series of optimizations:

$$(u^{k+1}, d^{k+1}) = arg\min_{u,d} J(u,d) - \langle p_u^k, u - u^k \rangle - \langle p_d^k, d - d^k \rangle + \frac{\lambda}{2} \|d - \phi(u)\|_2^2 \tag{4.20}$$

$$p_u^{k+1} = p_u^k - \lambda\big(\nabla\phi(u^{k+1})\big)\big(\phi(u^{k+1}) - d^{k+1}\big) \tag{4.21}$$

$$p_d^{k+1} = p_u^k - \lambda\big(d^{k+1} - \phi(u^{k+1})\big) \tag{4.22}$$

By the same simplification as in (4.11)-(4.17), the above iterations become

$$(u^{k+1}, d^{k+1}) = arg\min_{u,d} \|d\|_1 + E(u) + \frac{\lambda}{2} \left\|d - \phi(u) - b^k\right\|_2^2 \tag{4.23}$$

$$b^{k+1} = b^k + \big(\phi(u^{k+1}) - d^{k+1}\big) \tag{4.24}$$

To solve (4.23), ones can iteratively minimization with respect to u and d separately:

$$u^{k+1} = arg\min_u E(u) + \frac{\lambda}{2} \left\|d^k - \phi(u) - b^k\right\|_2^2 \tag{4.25}$$

$$d^{k+1} = arg\min_d \|d\|_1 + \frac{\lambda}{2} \left\|d - \phi(u^{k+1}) - b^k\right\|_2^2 \tag{4.26}$$

It is found that (4.26) can be easily solved using the shrinkage operator. The derivation is given in Appendix A. The optimal d of (4.26) can be computed as

$$d_j^{k+1} = shrink(\phi(u)_j + b_j^k, 1/\lambda) \tag{4.27}$$

for all j as the index of d^{k+1} where

$$shrink(x, \gamma) = \frac{x}{|x|} \cdot \max(|x| - \gamma, 0) \tag{4.28}$$

Computing the shrinkage operator is extremely fast which only requires a few operations. Solving (4.24) depends on the functional form of $E(u)$. Usually there is fast algorithm for solving it. The generalized algorithm is shown in Algorithm 4.1.

[20] mentioned that in many applications optimal efficiency is obtained when $N = 1$, i.e. only on iteration of inner loop is performed. Therefore, we keep $N = 1$ in all our split Bregman algorithms in this thesis.

4.4.2 Advantages of Split Bregman Iteration

As presented in the previous two sections, the original Bregman iteration is an improved version of traditional penalty function methods. One of the advantages over penalty function methods is that it has a fixed λ instead of getting a larger and larger λ. Thus the numerical instabilities that occur as $\lambda \to \infty$ are avoided.

Another advantage is that it converges very quickly for certain types of objective functions such as the l_1-regularized problem. As shown in Appendix A, the simple l_1-norm minimization

Algorithm 4.1 (Split Bregman Iteration)

INPUT: $E(u)$, $\phi(u)$, d
INITIALIZATION: $u^0 = 0$, $d^0 = 0$, $b^0 = 0$, $\lambda = 1$, $tol = 10^{-5}$
WHILE $\left\|u^k - u^{k+1}\right\|_2 > tol$ **DO**
INITIALIZATION: $\tilde{u}^0 = u^k$, $\tilde{d}^0 = d^k$
FOR $n = 1$ **TO** N **DO**

$$\tilde{u}^{n+1} \longleftarrow arg\min_u E(u) + \frac{\lambda}{2}\left\|\tilde{d}^n - \phi(u) - b^k\right\|_2^2$$

$$\tilde{d}^{n+1} \longleftarrow arg\min_d |d| + \frac{\lambda}{2}\left\|d - \phi(\tilde{u}^n) - b^k\right\|_2^2$$

END FOR
SET: $u^{k+1} = \tilde{u}^N$, $d^{k+1} = \tilde{d}^N$

$$b^{k+1} \longleftarrow b^k + \left(\phi(u^{k+1}) - d^{k+1}\right)$$

END WHILE
OUTPUT: solution u^* of optimization (4.18)

with the form

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|x - y\|_2^2 \tag{4.29}$$

has a simple closed form optimal solution given by the shrinkage operator since there is no coupling between entries of x. So the trick of split Bregman is to convert the optimization with coupled l_1-regularized term to the decoupled form (4.29). We will see several examples in the following section.

4.5 Fast l_1 Algorithms

In this section, we will derive fast algorithms for our l_1-norm minimization problem using the technique of split Bregman iterations. In the preceding two chapters, the l_1-regularized least squares (L1LS), the l_1-error and the l_1-regularized l_1-error (L1L1) have been introduced. Thus in the following their algorithms will be proposed respectively.

4.5.1 l_1-Regularized Least Squares

Recall in Chapter (2.4), we encounter the L1LS problem:

$$\min_x \mu \|Wx\|_1 + \frac{1}{2}\|y - Dx\|_2^2 \tag{4.30}$$

where μ is the weight that trades off the least square error and the sparsity of x and W is a diagonal positive-definite weighting matrix.

For convenience, we first consider the following formulation of L1LS to fit the setting of split Bregman formulation in (4.18):

$$\min_x \|Wx\|_1 + \frac{\lambda}{2}\|y - Dx\|_2^2 \tag{4.31}$$

By letting $\lambda = 1/\mu$, we make it equivalent to (4.30).

Then by operator splitting, (4.31) becomes

$$\min_{x,w} \|w\|_1 + \frac{\lambda}{2}\|y - Dx\|_2^2 \ s.t.\ w = Wx \tag{4.32}$$

Then we use Lagrange multiplier λ_1 to convert it into an unconstrained problem:

$$\min_{x,w} \|w\|_1 + \frac{\lambda}{2}\|y - Dx\|_2^2 + \frac{\lambda_1}{2}\|w - Wx\|_2^2 \tag{4.33}$$

Therefore, by split Bregman iterations,

$$(x^{k+1}, w^{k+1}) = arg\min_{x,w} \|w\|_1 + \frac{\lambda}{2}\|y - Dx\|_2^2 + \frac{\lambda_1}{2}\left\|w - Wx - b^k\right\|_2^2 \tag{4.34}$$

$$b^{k+1} = b^k + (Wx^{k+1} - w^{k+1}) \tag{4.35}$$

Then we iteratively solve the minimization (4.34),

$$x^{k+1} = arg\min_{x} \frac{\lambda}{2}\|y - Dx\|_2^2 + \frac{\lambda_1}{2}\left\|w^k - Wx - b^k\right\|_2^2 \tag{4.36}$$

$$w^{k+1} = arg\min_{w} \|w\|_1 + \frac{\lambda_1}{2}\left\|w - Wx^{k+1} - b^k\right\|_2^2 \tag{4.37}$$

For (4.36), the solution can be easily obtained since it is differentiable:

$$x^{k+1} = (\lambda D^T D + \lambda_1 W^T W)^{-1}\left(\lambda D^T y + \lambda_1 W^T (w^k - b^k)\right) \tag{4.38}$$

Then similar to (4.26), (4.37) can be solved by shrinkage operator for all entries j:

$$w_j^{k+1} = shrink\left((Wx^{k+1})_j + b^k, 1/\lambda_1\right) \tag{4.39}$$

where the shrinkage operator is defined as (4.27). Therefore, the algorithm of L1LS can be formulated as shown in Algorithm 4.2.

4.5.2 l_1-Error

Next we consider the l_1-error problem (3.3),

$$\min_{x} \|e\|_1 \ \ s.t.\ e = y - Dx \tag{4.40}$$

By Lagrange multiplier, (4.40) becomes

$$\min_{x,e} \|e\|_1 + \frac{\lambda}{2}\|e - (y - Dx)\|_2^2 \tag{4.41}$$

Algorithm 4.2 (l_1**-regularized least squares)**

INPUT: W, D, λ, y
INITIALIZATION: $u^0 = 0$, $d^0 = 0$, $b^0 = 0$, $\lambda_1 = 1$, $tol = 10^{-5}$
WHILE $\left\|x^k - x^{k+1}\right\|_2 > tol$ **DO**

$$x^{k+1} \longleftarrow (\lambda D^T D + \lambda_1 W^T W)^{-1}\left(\lambda D^T y + \lambda_1 W^T (w^k - b^k)\right)$$

FOR ALL entry j DO

$$w_j^{k+1} \longleftarrow shrink\left((Wx^{k+1})_j + b^k, 1/\lambda_1\right)$$

END FOR

$$b^{k+1} \longleftarrow b^k + (Wx^{k+1} - w^{k+1})$$

END WHILE
OUTPUT: solution x^* of optimization (4.31)

Using split Bregman, (4.41) can be separated in the following iterative minimization subproblems:

$$x^{k+1} = arg\min_x \left\|Dx + e^k - y - b^k\right\|_2^2 \tag{4.42}$$

$$e^{k+1} = arg\min_e \|e\|_1 + \frac{\lambda}{2}\left\|e - (y - Dx^{k+1} - b^k)\right\|_2^2 \tag{4.43}$$

$$b^{k+1} = b^k + \left((y - Dx^{k+1}) - e^{k+1}\right) \tag{4.44}$$

The solution of (4.43) and (4.44) can be easily obtained:

$$x^{k+1} = (D^T D)^{-1} D^T (y - e^k + b^k) \tag{4.45}$$

$$e_j^{k+1} = shrink\left((y - Dx^{k+1})_j + b_j^k, 1/\lambda\right), \forall j \tag{4.46}$$

The algorithm is shown in Algorithm 4.3.

Algorithm 4.3 (l_1**-error**)

INPUT: D, y
INITIALIZATION: $x^0 = 0$, $e^0 = 0$, $b^0 = 0$, $\lambda = 1$, $tol = 10^{-5}$
WHILE $\left\|x^k - x^{k+1}\right\|_2 > tol$ **DO**

$$x^{k+1} \longleftarrow (D^T D)^{-1} D^T (y - e^k + b^k)$$

FOR ALL entry j DO

$$e_j^{k+1} \longleftarrow shrink\big((y - Dx^{k+1})_j + b_j^k, 1/\lambda\big)$$

END FOR

$$b^{k+1} \longleftarrow b^k + \big((y - Dx^{k+1}) - e^{k+1}\big)$$

END WHILE
OUTPUT: solution x^* of optimization (4.40)

4.5.3 l_1-Regularized l_1-Error

In Chapter 3, the L1L1 minimization problem was introduced as

$$\min_{x,e} \|e\|_1 + \mu \|Wx\|_1 \ \ s.t.\ e = y - Dx \tag{4.47}$$

Let $\tilde{W} = \mu W$, (4.47) can be transformed to an unconstrained optimization using Lagrange multiplier:

$$\min_{x,e,w} \|e\|_1 + \|w\|_1 + \frac{\lambda_1}{2} \|e - (y - Dx)\|_2^2 + \frac{\lambda_2}{2} \left\|w - \tilde{W}x\right\|_2^2 \tag{4.48}$$

Using split Bregman iteration, we separate this optimization

problem into many smaller subproblems:

$$x^{k+1} = arg\min_x \frac{\lambda_1}{2}\left\|e-(y-Dx)-b_e^k\right\|_2^2 + \frac{\lambda_2}{2}\left\|w-\tilde{W}x-b_w^k\right\|_2^2 \tag{4.49}$$

$$e^{k+1} = arg\min_e \|e\|_1 + \frac{\lambda_1}{2}\left\|e-(y-Dx^{k+1})-b_e^k\right\|_2^2 \tag{4.50}$$

$$w^{k+1} = arg\min_w \|w\|_1 + \frac{\lambda_2}{2}\left\|w-\tilde{W}x^{k+1}-b_w^k\right\|_2^2 \tag{4.51}$$

$$b_e^{k+1} = b_e^k + (y-Dx^{k+1}-e^{k+1}) \tag{4.52}$$

$$b_w^{k+1} = b_w^k + (\tilde{W}x^{k+1}-w^{k+1}) \tag{4.53}$$

The solutions of (4.49)-(4.51) will be:

$$x^{k+1} = (\lambda_1 D^T D + \lambda_2 \tilde{W}^T\tilde{W})^{-1}\left[-\lambda_1 D^T(e^k-y-b_e^k)+\lambda_2\tilde{W}^T(w^k-b_w^k)\right] \tag{4.54}$$

$$e_j^{k+1} = shrink\big((y-Dx^{k+1})_j + (b_e^k)_j, 1/\lambda_1\big), \forall j \tag{4.55}$$

$$w_j^{k+1} = shrink\big((\tilde{W}x^{k+1})_j + (b_w^k)_j, 1/\lambda_2\big), \forall j \tag{4.56}$$

The generalized algorithm of L1L1 is presented in Algorithm 4.4.

4.6 Summary

In this chapter, we have derived all the l_1 algorithms we need for our face alignment task using the split Breman. These algorithms have a common characteristic: they all converge very fast, and can be computed with just a few simple operations. Actually, there are other fast l_1 algorithms including the gradient projection methods [19], homotopy methods [29], iterative shrinkage-thresholding methods [13], proximal gradient methods

Algorithm 4.4 (l_1-regularized l_1-error)

INPUT: μ, W, D
INITIALIZATION: $x^0 = 0$, $e^0 = 0$, $b^0 = 0$, $\lambda_1 = \lambda_2 = 1$, $tol = 10^{-5}$
WHILE $\left\|x^k - x^{k+1}\right\|_2 > tol$ **DO**

$$x^{k+1} \longleftarrow (D^T D)^{-1} D^T (y - e^k + b^k)$$

FOR ALL entry j DO

$$e_j^{k+1} \longleftarrow shrink\big((y - Dx^{k+1})_j + (b_e^k)_j, 1/\lambda\big)$$

END FOR
FOR ALL entry j DO

$$w_j^{k+1} \longleftarrow shrink\big((\tilde{W}x^{k+1})_j + (b_w^k)_j, 1/\lambda_2\big)$$

END FOR

$$b_e^{k+1} \longleftarrow b_e^k + \big((y - Dx^{k+1}) - e^{k+1}\big)$$

$$b_w^{k+1} \longleftarrow b_w^k + (\tilde{W}x^{k+1} - w^{k+1})$$

END WHILE
OUTPUT: solution x^* and e^* of optimization (4.47)

[30], and augmented Lagrange multiplier methods (ALM) [49], which are discussed in a review paper by Yang et al. [48]. They compare different algorithms using the application for robust face recognition. The L1L1 problem is also formulated using those algorithms. The conclusion is that there is no definite winner that always gives the best performance in terms of both accuracy and speed. In their application, Homotopy and ALM were found to achieve the highest recognition rates and the lowest computational cost among the l_1 algorithms discussed. There are papers [50, 38] discussing the equivalence between Bregman iteration and ALM when the constraints are all linear. However this is out of the scope of our thesis.

Chapter 5

Face Alignment Using Sparse and Low-rank Decomposition

5.1 Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images (RASL)

In previous chapters, we have prepared all the algorithms necessary for our proposed face alignment method. So starting from this chapter, we will concentrate on the problem of face alignment. To begin with, a pixel-based batch alignment method called Robust Alignment by Sparse and Low-rank Decomposition for Linearly Correlated Images (RASL) is explained. RASL is able to simultaneously align a batch of linearly correlated images such as face images despite occlusion or any kinds of gross corruption. It seeks an optimal set of image domain transformations such that the matrix of transformed images can be decomposed as the sum of a low-rank matrix of aligned images and a sparse matrix of errors. Its principle is based on the knowl-

edge that if the images are well-aligned, they should show good low-rank structure up to some sparse corruptions. Face images are among the group of linearly correlated images which possess this property.

During the decomposition of the matrix into a low-rank and a sparse matrix, RASL applies the technique of principal component pursuit introduced in Chapter 3 since the original rank minimization problem is non-convex. For face alignment, we would like to find a set of transformation that allows us to transform each image to the same pose. For simplicity, we consider that the transformation is 2-D affine transform. i.e. we implicitly assume the face of an individual is approximately on a plane in 3-D space. Therefore, the idea of RASL is to search for a set of 2-D affine transformation parameters τ such that the rank of the transformed images becomes as small as possible and at the same time the sparse errors are compensated.

5.2 Problem Formulation

This section introduced the whole formulation of RASL, by presenting two main concepts, convex relaxation and iterative linearization. Then the algorithm is given as a conclusion.

5.2.1 Theory

Given $I_1, \cdots, I_n \in \mathbb{R}^{w \times h}$ as the original misaligned grayscale images of a person's face. Define $vec : \mathbb{R}^{w \times h} \mapsto \mathbb{R}^m$ as the operator that selects an m-pixel region of interest (e.g. the face part with

main features such as eyes, nose and mouths) from an image and stacks to be a vector. Denote $\tau = \{\tau_1, \ldots, \tau_n\}$ as the set of transformation, and $D \circ \tau$ as shorthand for $[vec(I_1 \circ \tau_1)| \ldots |vec(I_n \circ \tau_n)] \in \mathbb{R}^{m \times n}$, where $I \circ \tau$ represents image I after transformed by τ. The problem is formulated as the minimization in Lagrangian form:

$$\min_{A,E,\tau} rank(A) + \gamma \|E\|_0 \ s.t. \ D \circ \tau = A + E \qquad (5.1)$$

Here, the $\|\cdot\|_0$ represents the number of nonzero entries in the error matrix E, and $\gamma > 0$ controls the weighting between the rank of solution and the sparsity of the error.

The optimization in (5.1) is not directly tractable: both rank and l_0-norm are non-convex and discontinuous and the equality constraint $D \circ \tau = A + E$ is nonlinear. [32] introduces two techniques, called the convex relaxation and the iterative linearization.

Convex Relaxation

The convex relaxation involves the replacement of $rank(\cdot)$ and $\|\cdot\|_0$ with the sum of the singular values $\|A\|_* \doteq \sum_{i=1}^{m} \sigma_i(A)$, namely the nuclear norm, and the l_1-norm $\|E\|_1 \doteq \sum_{i=1}^{m} |E_{ij}|$ respectively. The problem (5.1) becomes:

$$\min_{A,E,\tau} \|A\|_* + \lambda \|E\|_1 \ s.t. \ D \circ \tau = A + E \qquad (5.2)$$

According to Section 3.3.2 in Chapter 3, it is wise to choose the weighting parameter λ to be in the form of $C/\sqrt{m}$ where C is a constant typically set to 1.

Iterative Linearization

For the nonlinear constraint $D \circ \tau = A + E$, we can approximate it by linearizing about the current estimate of τ^0 for small change of τ. Then (5.2) can be written as:

$$\min_{A,E,\Delta\tau} \|A\|_* + \lambda \|E\|_1 \ \ s.t.\ D \circ \tau^0 + \sum_{i=1}^{n} J_i \Delta\tau \epsilon_i^T = A + E \quad (5.3)$$

where $J_i \doteq \frac{\partial}{\partial \zeta} vec(I_i \circ \zeta)|_{\zeta=\tau_i}$ is the Jacobian of the i-th image with respect to the transformation parameters τ_i, $\tau = [\tau_1 | \cdots | \tau_n]$ and ϵ_i denotes the standard basis for $\mathbb{R}^n$.

Notice that the linearization only holds locally, so the solution of (5.3) $\tau + \Delta\tau$ may not exactly solve (5.2). In order to find the minimum of (5.2). $\Delta\tau$ has to be iteratively solved by (5.3) using the current estimate of τ for each iteration. [32] has shown that as long as the initial misalignment is not too large, the solution after the iterations effectively recovers the correct transformations τ which separates the low-rank structure of the aligned images from any sparse errors.

5.2.2 Algorithm

The algorithm of RASL is shown in Algorithm 5.1. It involves two loops of iterations. The outer loop corresponds to the iterations of (5.3) which updates τ. The inner loop is the algorithm for solving the linearized convex optimization in step 3 of Algorithm 5.1. To solve it, [32] suggests to use the Accelerated Proximal Gradient algorithm. The algorithm can be implemented according to [32].

Algorithm 5.1: RASL [32]

INPUT: Image $I_1, \ldots, I_n \in \mathbb{R}^{w \times h}$, initial transformation $\tau_1, \ldots, \tau_n$ in affine group, weight $\lambda > 0$

WHILE not converged **DO**

Step 1: compute Jacobian matrices w.r.t. transformation:

$$J_i \longleftarrow \frac{\partial}{\partial \zeta} \left(\frac{vec(I_i \circ \zeta)}{\|I_i \circ \zeta\|_2} \right) \Big|_{\zeta=\tau_i}, \ i = 1, \ldots, n$$

Step 2: warp and normalize the images:

$$D \circ \tau \longleftarrow \left[\frac{vec(I_1 \circ \tau_1)}{\|vec(I_1 \circ \tau_1)\|_2} | \cdots | \frac{vec(I_n \circ \tau_n)}{\|vec(I_n \circ \tau_n)\|_2} \right]$$

Step 3 (inner loop): solve the linearized l_1-LS:

$$(A^*, E^*, \Delta\tau^*) \longleftarrow$$

$$arg \min_{A,E,\Delta\tau} \|A\|_* + \lambda \|E\|_1 \ s.t. \ D \circ \tau + \sum_{i=1}^{n} J_i \Delta\tau_i e_i^T = A + E$$

Step 4: update transformation:

$$\tau \longleftarrow \tau + \Delta\tau^*$$

END WHILE

OUTPUT: solution A^*, E^*, τ of optimization (5.2)

5.3 Direct Extension of RASL: Multi-RASL

One notable limitation of RASL appears when the image size is large and the amount of images is large. For the batch alignment, all the images have to stack into the memory to perform the computation. If the amount as well as the size of the images are large, memory might not be able to store all the data during computation. So the original RASL is not capable for large-scale problem.

5.3.1 Formulation

We propose here a direct extension of RASL. Instead of aligning all images in one batch, we divide into smaller batches, so that for each batch the memory is enough for the computation. However, if we just do RASL as in Section 5.3, every batch would have a different alignment. Therefore, a reference image is chosen to be our ground pose, that is, all the images are aligned according to the pose in the reference image. It can be achieved by adding the reference image to each batch before applying RASL. Our proposed method is named Multi-RASL because RASL has to be performed many times separately for every batch.

The key of our algorithm is that within each batch, all images are supposed to be well-aligned and the transformation parameters τ from original to the well-aligned pose are computed. Thus we get the inverse transform for the reference image to transform back to the original pose. By applying such an inverse transformation to all images within the batch, in principle, all the

images can be aligned to the reference. And same principle can be applied to all batches. Eventually all images in the dataset can be aligned.

The inverse transform can be explained as follows: Input one batch of images together with the reference into Algorithm 5.1 in Section 5.2.2. Then we can get the transform parameters τ for each image.

Let τ_{ref} be the transform parameters from misaligned I_{ref} to aligned $\bar{I}_{ref}$, i.e. $I_{ref} \circ \tau_{ref} = \bar{I}_{ref}$. So in order to align all the images according the reference pose, we apply the inverse transform τ_{ref}^{-1} on the aligned images. The process can be represented as follow:

$$\bar{\bar{I}}_i = \bar{I}_i \circ \tau_{ref}^{-1} \tag{5.4}$$

$$= (I_i \circ \tau_i) \circ \tau_{ref}^{-1} \tag{5.5}$$

$$= I_i \circ (\tau_i \circ \tau_{ref}^{-1}) \tag{5.6}$$

Thus, $\tilde{\tau}_i \doteq \tau_i \circ \tau_{ref}^{-1}$ is the transform parameters from misaligned I_i to aligned with reference pose $\bar{\bar{I}}_i$, since the transformation is associative.

5.3.2 Algorithm

The algorithm of Mulit-RASL is basically the same as that of RASL, except there are three extra steps, choosing the reference, dividing dataset into partitions and performing an inverse transformation, respectively.

In choosing the reference, one can either directly select from the dataset, or pick an aligned image resulting from RASL on

a uniformly random small subset of the dataset. Empirically, the latter usually gives you a better choice if you would like to align all the face photos to the frontal normal view. However, Multi-RASL is not restricted to the frontal normal view.

The objective of dividing the dataset into small batches is to ensure the memory is enough to go through the RASL computation for each batch. In practice, the alignment would never be perfect, thus there will be always some errors in the alignment results although it may not be too large. Even though we have set a reference image to reduce the difference in alignment among the batches, large alignment difference between batches can still occur. Therefore, it is wise to choose as small as possible the number of partitions such that in each partition the memory is enough for computation. For simplicity, we prefer equal partition.

After getting the transformation parameters τ for every image by RASL, we apply the inverse transform method in (5.4) to align all the images to reference pose. The whole algorithm is summarized in Algorithm 5.2.

5.4 Matlab Implementation Details

After discussing the algorithm, this section will discuss the practical implementation of it using Matlab. Matlab is universal mathematical software for doing simulations. It is particularly useful in simulating image processing problems as it has a fully developed image processing toolbox which contains a wide variety of image processing functions such as the one for affine

Algorithm 5.2: Multi-RASL

INPUT: Image $I_1, \ldots, I_n \in \mathbb{R}^{w \times h}$, initial transformation $\tau_1, \ldots, \tau_n$ in affine group, weight λ

Step 1: Divide Image $I_1, \ldots, I_n$ into K small groups (also for initial transformation $\tau_1, \ldots, \tau_n$), each group has approximately equal amount of images

Step 2: Select a reference by either directly from the dataset, or from result of RASL on a uniformly random small subset of the dataset

Step 3: FOR ALL BATCHES, go through Algorithm 5.1 respectively

Step 4:
FOR ALL i from 1 to n **DO**

$$\tilde{\tau}_i \longleftarrow \tau_i \circ \tau_{ref}^{-1}$$

END FOR

OUTPUT: Transformation parameters $\tilde{\tau}_1, \ldots, \tilde{tau}_n$

transform of 2-D images. Our algorithms can be easily implemented using Matlab. Thus we choose it to be our simulation platform.

Basically, the implementation of Mulit-RASL is divided into:

1. Obtain the image sequence. If the image resolution is high, scale down the image to a reasonable resolution.
2. Detect the faces in the images and cut them out. Save the position of face on each image so as to compute the overall affine transformation matrix for the whole image.
3. Select a reference either directly from the dataset, or from result of RASL on a uniformly random small subset of the dataset.
4. Go through the multi-RASL algorithm. Then the transformation parameters for the cut-out faces can be obtained.
5. Combine the transformation parameters for the cut-out faces together with the transformation matrix due to downsampling and position of face to calculate the overall transformation matrix for every image (in original resolution)
6. Finally, transform the original image sequence using the overall transformation matrices calculated.

5.4.1 Preprocessing

In order to start our face alignment, firstly a video sequence of a person's face is collected. The video can be an interview of a person, or a person giving a speech. It can be in different

formats, for example, MPEG (.mpg), AVI (.avi) or Flash Video (.flv), etc. We apply some video processing software to convert the video into a set of images so that we can perform our computation on it. For simplicity, we assume there is only one person's face on each frame.

Sometimes the image resolution is too high which may lower the efficiency of our algorithm. In fact, so long as the image is recognizable by human, its resolution is sufficient for a reasonable accurate alignment. Therefore, for high resolution images, it is wise to down-sample them.

Usually, the face may not stay in a fixed position. Also, there may be some camera effects such as zooming in and out, which causes the changing in size of the face. Thus, some techniques have to be applied in order to capture the face image out so as to align them. Two techniques have been tested for this preprocessing step, including face detection and 3-D pose tracking.

Face Detection

We apply the available face detection algorithm for tracking the face in our video sequence. There are many off-the-shelf robust and fast face detection algorithms using various approaches such as neural networks and weak classifier. In our application, it is sufficient to have just a rough estimation of the position of face. So for convenient, we pick a face detector running on Matlab platform [24]. Figure 5.1 demonstrates the output of face detection algorithm. It locates the position of a human face.

Figure 5.1: Output of face detection algorithm

3-D Pose Tracking

In our introduction of face alignment algorithm in Chapter 1.1.3, some 3-D face model alignment methods have been introduced. We choose one of the methods called 3-D model-based pose tracking to be one choice for our preprocessing. One advantage of using results from 3-D pose tracking as the input is that it increases the speed of convergence of RASL and hence lowers the computational cost since the input has already approximately aligned. Figure 5.2 shows the process of 3-D pose tracking[1].

Figure 5.2: 3-D pose tracking

[1]The code for 3-D pose tracking is provided by Microsoft Research Asia.

5.4.2 Transformation

In Matlab Image Processing Toolbox, there are well-defined functions for performing the image transformation. We apply two of the major functions for our affine transform. First, the function *maketform*() is used to create spatial transformation structure (TFORM). We choose the transform type as "affine". Next, the function *imtransform*() is utilized for 2-D spatial transformation to image given the affine transformation structure TFORM generated in the previous step.

The function *imtransform* employs the inverse mapping technique. Instead of choosing the forward mapping which may cause gaps and overlapping in the output space, it is more stable to start from the output space. Let the position of the k-th output pixel be (x_k, y_k). The image transform procedure can be described as follows:

1. Set a range for the output image in output space.
2. Within the range of output, pick an output image pixel, say (x_k, y_k).
3. Apply the inverse spatial transform T^{-1} to determine the corresponding location in input space: $(u_k, v_k) = T^{-1}\big((x_k, y_k)\big)$. In our application, T is the defined affine transform using *maketform*.
4. Using the input image pixels nearest to (u_k, v_k), interpolate to get an approximation value for (u_k, v_k). Here we choose bicubic interpolation.

5. Set the approximation value for the output pixel (x_k, y_k).

5.4.3 Jacobian J_i

In step 1 of Algorithm 5.1, the Jacobian matrices corresponding to the transformation τ_i for $i = 1, \ldots, n$ is computed as

$$J_i \longleftarrow \frac{\partial}{\partial \zeta} \left(\frac{vec(I_i \circ \zeta)}{\|I_i \circ \zeta\|_2} \right) \Bigg|_{\zeta=\tau_i} \tag{5.7}$$

However in practice, Jacobian J_i cannot directly computed using (5.7). The following describes the procedure for obtaining the Jacobian. For simplicity, we omit the subscript so that J represents the Jacobian matrix for a single image. Here the image is expressed as a column vector I in which each entry represents one pixel of the image. Also, since there are 6 parameters for affine transformation τ which can use the affine transformation matrix α to represent. The affine transformation can be written as

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \alpha_4 & \alpha_5 & \alpha_6 \end{bmatrix}$$

Next we let $\tau = [\alpha_1 \ \alpha_2 \ \alpha_3 \ \alpha_4 \ \alpha_5 \ \alpha_6]^T$. Then let us consider the iterative linearization where the Jacobian appears:

$$I \circ \tau = I \circ (\tau^0 + \Delta\tau) \approx I \circ \tau^0 + J\Delta\tau \tag{5.8}$$

So we defined $\Delta I \doteq J\Delta\tau$. In order to get Jacobian J, we consider each entry of ΔI can be represented

$$\Delta I_i = \nabla_x I_i \Delta x + \nabla_y I_i \Delta y = [\nabla_x I_i \ \nabla_y I_i] \, \Delta p \tag{5.9}$$

where $\Delta p = [\Delta x \ \Delta y]$ and $\nabla_x I_i$ and $\nabla_y I_i$ are the gradient of image intensity at each pixel in x and y direction respectively.

These gradients can be computed using Sobel operator. There is corresponding function in Matlab. Here Δx and Δy are the change in position due to the image transformation $\Delta\tau$ which is assumed to be very small. Then for affine transform, we have the following relation:

$$\Delta p = J_{aff}\Delta\tau \tag{5.10}$$

where $J_{aff} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$ which is the Jacobian for position change in terms of transformation matrix α and $\Delta\tau = [\Delta\alpha_1\ \Delta\alpha_2\ \Delta\alpha_3\ \Delta\alpha_4\ \Delta\alpha_5\ \Delta\alpha_6]^T$.

Thus we can obtain the Jacobian for each pixel by computing

$$[\nabla_x I_i\ \nabla_y I_i]\, J_{aff} = [\nabla_x I_i x\ \nabla_x I_i y\ x\ \nabla_y I_i x\ \nabla_y I_i y\ y] \tag{5.11}$$

which is the i^{th} row of the Jacobian J.

5.5 Experiments

In this section two experiments have been performed to demonstrate the capability and efficacy of RASL and our extension Multi-RASL. First the performance of RASL and Multi-RASL is compared using a small dataset with 100 images. The reason for using a small dataset is that we want to show the performance of Multi-RASL is comparable with that of RASL. Then for large dataset where ordinary RASL fails, we evaluate our proposal quantitatively for different division of batches. The eyebrow corners have been extracted and used for our comparison.

5.5.1 Qualitative Evaluations Using Small Dataset

In this test, a dataset with total number of images $N = 100$ of the same person is obtained from an interview video collected from the internet[2]. The original dimension of each images is 344×260. Since the size of the faces from distinct frames may be different due to zooming effect, it is suitable to transform all the faces into the canonical frame for computation. We set the size of face in canonical frame to be 65×75. First we perform the 3D pose tracking as pre-processing. Therefore our method can be viewed as an enhancement of 3D pose tracking.

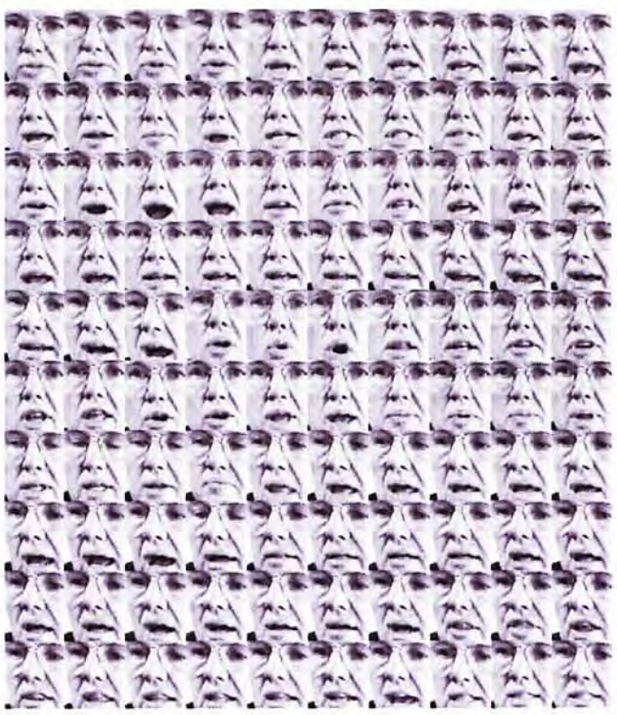
RASL

Figure 5.3 displays the output of RASL. This includes the original images D after 3D pose tracking, aligned images $D \circ \tau$, low-rank component A and sparse corruption E respectively. Figure 5.3b is decomposed into the low-rank matrix (Figure 5.3c) and sparse matrix (Figure 5.3d). Notice that Figure 5.3d captures the mouth part due to movement and also the eye part for closed eyes. This validates our assumption that the moving mouth and blinking eyes can be considered as sparse noise or corruption. Comparing Figure 5.3a and 5.3b, it is obvious that RASL improves over the results of 3D pose tracking.

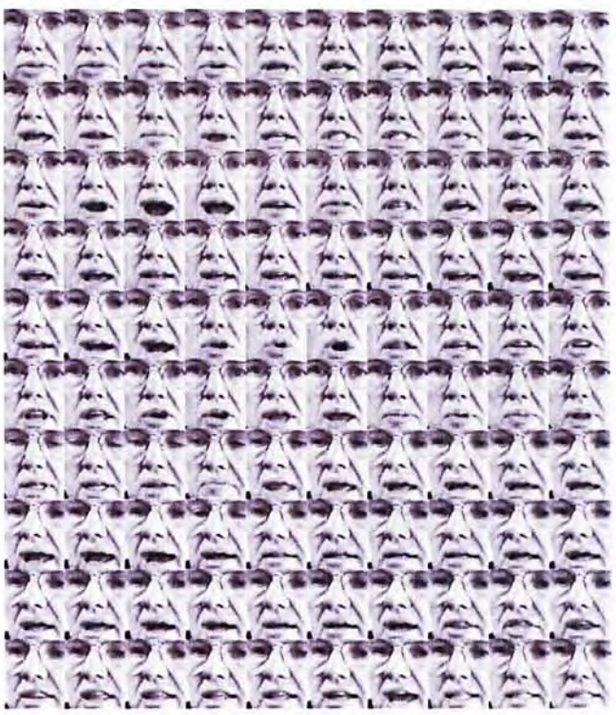
Multi-RASL

Now divide the original 100 images into two batches, each with 50 images. In order to apply our Multi-RASL algorithm, we

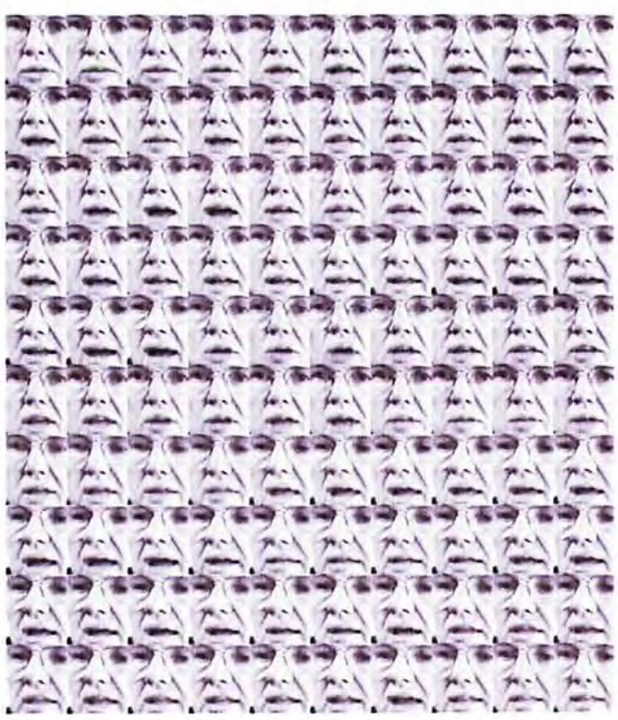
[2]The video is collected from `http://www.beet.tv/2008/09/microsofts-crai.html`.



(a) Original images D



(b) Aligned images $D \circ \tau$



(c) Low-rank component A

(d) Sparse corruption E

Figure 5.3: Output of RASL

choose the reference from one of RASL results from Figure 5.3b for a fair comparison. The chosen reference is shown in Figure 5.4. The input is given in Figure 5.5.

Figure 5.4: Reference image for Multi-RASL

We test the two situations, Multi-RASL with 2 batches and 10 batches. In each case, the amount of images in all batches is equally divided by splitting the temporal video sequence. Figure 5.6a and 5.6b show the output for cases with 2 batches and 10 batches respectively. For comparisons, we put together the results of RASL in colour in Figure 5.6c.

The above experiment shows Multi-RASL algorithms can achieve comparable quality of alignment as RASL. Besides, we can fix the reference pose such that all the images in the dataset are aligned to it. Careful inspection would discover that the alignment using 10 batches is not as good as that using 2 batches. Although the difference is small, it is still noticeable. The more of the batches the more misalignment observes among different batches.

Memory and Speed

Our experiment are carried out on a 2.66GHz Intel Core 2 Quad machine with 3.25GB RAM and 32-bit Operating System.

Figure 5.5: Input images

Algorithm	Approximate Time required
RASL	210s
Multi-RASL (2 batch)	100s
Multi-RASL (10 batch)	60s

Table 5.1: Approximate running time for 100 images in RASL and Multi-RASL experiment

(a) Multi-RASL with 2 batches

(b) Multi-RASL with 10 batches

(c) Ordinary RASL

Figure 5.6: Alignment output using Multi-RASL and RASL

The problem of limited memory always exists due to physical constraints. RASL requires to store all N images in memory during alignment. It may be practical for hundreds of images, but not for thousands or millions of images. In comparison, our extension algorithm requires to store only the number of images in each batch at one time, thus the use of RASL is possible.

The speed of RASL depends much on the number of outer loop iterations as the inner loop algorithm for sparse and low rank decomposition is extremely slow because of SVD. For images with large misalignments, the time required would increase significantly as more outer loop iterations are needed. The approximate running time for the above test is shown in Table 5.1. It is found that the time require for more batches is less. It may be due to our temporal splitting, so within each batch, the face pose is similar. Hence it is easy for all images in the batch to converge to one aligned pose, providing a fast speed of convergence. This reveals a tradeoff between quality and speed of Multi-RASL.

5.5.2 Large Dataset Test

In this test, we use the video of Oval Office Address of the US President Obama[3]. 5000 consecutive frames with dimension 280×270 were chosen to be the sample dataset of our experiment. This time we apply the off-the-shelf face detector [24] as preprocessing. We set the size of face in canonical frame to be 60×65. We follow the same stopping criteria as in previous

[3]The video is collected from `http://www.youtube.com/watch?v=Gh76oepKFc8`.

experiment.

Since the amount of images has exceeded the limit that RASL can handle, the only way to deal with this large-scale problem by sparse and low-rank decomposition is to use Multi-RASL. Three Multi-RASL tests have been performed using different number of batches.

Figure 5.7 shows the reference image. It is chosen from output of RSAL on 100 uniform samples of the 5000 images.

Figure 5.7: Reference image for Multi-RASL

Quantitative Evaluation: Eyebrow Corners Positions

In order to have a quantitative evaluation, we have to select a fair feature on the face so as to obtain the statistics of errors for comparison. The eye corners have been chosen as the features in [47, 32]. However, in our dataset the eyes are not always open. It is impossible to obtain the eye corners in our case. We have assumed the face is a flat plane, so it is not suitable to use the nose position to evaluate the alignment as it is definitely not on the plane. Besides, the mouth is moving causing difficulties in getting fair features points for our test. Hence we decide to get the eyebrow corners positions as our fair features

since the shapes of the eyebrows on 3D faces basically remain unchanged no matter what kind of movement due to the eyes and mouths. Also, the eyebrows corners are hardly displaced even under different emotional expressions.

To fairly compare using distances in pixels, we first go through a normalization process. The distance between left and right eyebrow corners are normalized to 50 pixels. Since all the face images are expected to align according to the reference, so we define the distance between the estimated eyebrow corners of the aligned face and the reference face to be the error. Table 5.2 shows the statistics of the errors in eyebrow corners using the three approaches in our experiment. To make some sense to the numbers, we include also the statistics of the errors for the output of RSAL on 100 uniform samples of the 5000 images in Table 5.3. This time there is no specific reference for the 100 samples, thus we calculate the error by measuring the distance from the estimated eyebrow corners to their statistical center.

Result reveals that using fewer batches allows more accurate alignments. However, comparing with RASL for 100 images, the errors are still far too large (more that one pixel).

Qualitative Evaluation

To illustrate the effectiveness of Multi-RASL, using just the eyebrow corners is not enough. We provide in Figure 5.8 some of the alignment output using the three different Multi-RASL. We pick one frame from every 50 consecutive frames so that at least one image from each batch is included, with a total 100 frames. For

		Left eyebrow	Right eyebrow	Average
Mean error	(a)	2.1690	3.6629	2.9160
	(b)	2.4742	3.7808	3.1275
	(c)	2.7513	3.8790	3.3152
Standard error	(a)	1.1367	1.7701	1.4534
	(b)	1.3352	1.8133	1.5743
	(c)	1.4223	1.8377	1.6300
Maximum error	(a)	6.1304	8.2817	7.2061
	(b)	6.1320	8.9388	7.5354
	(c)	6.1345	8.9425	7.5535

Table 5.2: Eyebrow corners comparison of Multi-RASL with (a) 25 batches, (b) 50 batches and (c) 100 batches. Here the distances are measured from the estimated eyebrow corners on faces to that of reference.

	Left eyebrow	Right eyebrow	Average
Mean error	1.5635	1.8973	1.7304
Standard error	0.7698	0.8791	0.8245
Maximum error	3.9696	3.5957	3.7827

Table 5.3: Eyebrow corners comparison of RASL on 100 images. Here the distances are measured from the estimated eyebrow corners to their center.

the three Multi-RASL outputs, the eyebrow corners are marked in white for better comparison. The 25-batch approach gives the best alignment which is consistent with the quantitative evaluation.

Computational Time

Our experiment are carried out on a 2.66GHz Intel Core 2 Quad machine with 3.25GB RAM and 32-bit Operating System. The approximate CPU time for the 5000 images dataset for different Multi-RASL is given in Table 5.4. It shows that the 100-batch approach runs the fastest.

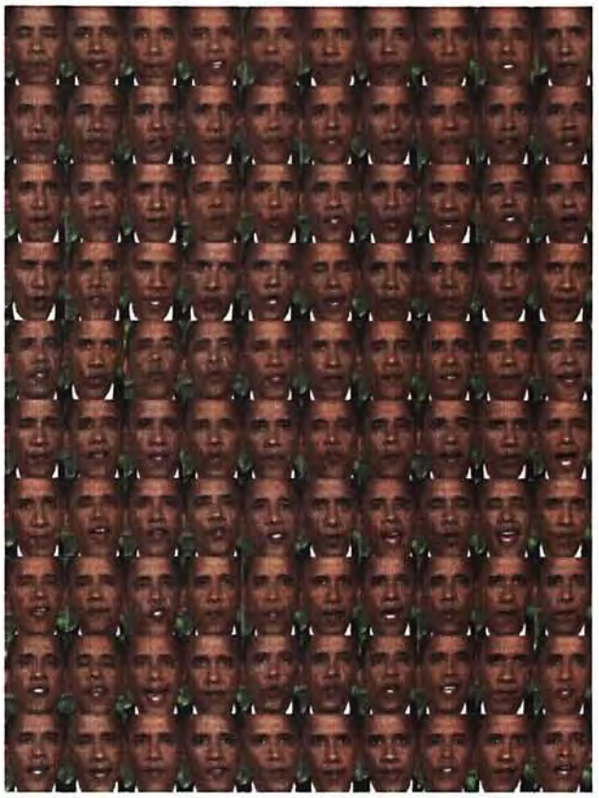
Algorithm	Approximate Time required
Multi-RASL (25 batch)	17180s
Multi-RASL (50 batch)	10660s
Multi-RASL (100 batch)	6680s

Table 5.4: Approximate running time for 5000 images in Multi-RASL experiment

5.5.3 Conclusion

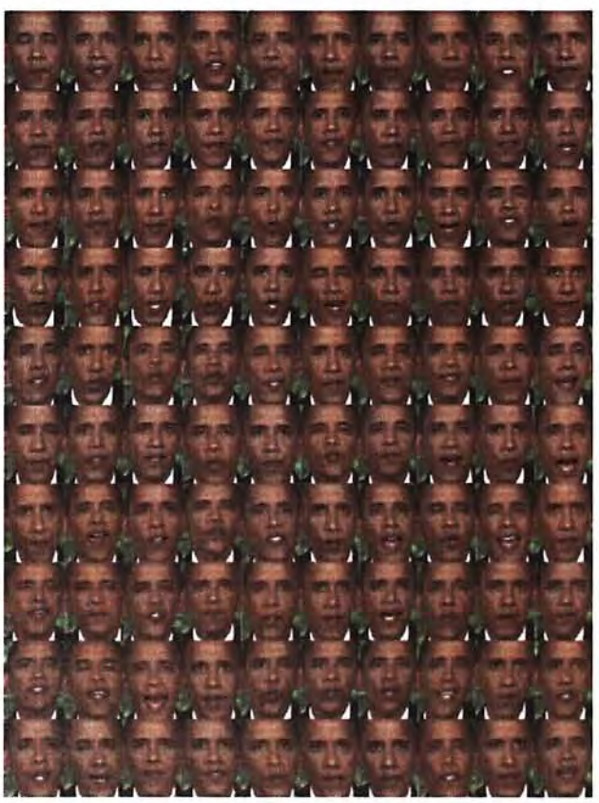
Experiment results have shown that our Multi-RASL algorithm gives a comparable alignment output as RASL. However, as seen in Figure 5.8, using more batches may cause a significant misalignment across different batches. One reason may due to the inaccurate of our assumption that the face is the flat plane in 3D space. This assumption is generally false and hence it introduces errors in the alignment. Within each batch, the error is relatively small and thus hardly noticeable. But for different

(a) Original images



(b) Multi-RASL with 25 batches

(c) Multi-RASL with 50 batches



(d) Multi-RASL with 100 batches

Figure 5.8: 100 uniformly sampled images (with eyebrow corners marked) to illustrate efficiency of Multi-RASL

batches, since the alignments are independent, the alignment errors become significant. It concludes that it is wise to use as few batches as possible for a better alignment result.

Unfortunately, using fewer batches usually lead to a longer running time. As shown in Table 5.1 and 5.4, there may be a tradeoff between the accuracy and speed. Therefore, in practice, the number of batches would depend on the condition and requirement of the specific task.

5.6 Sensitivity analysis on selection of references

One important difference of Multi-RASL from RASL is that we have to choose a reference. In our application of photo-realistic talking head, the frontal normal view is the most useful pose. Therefore, we would like to choose the best frontal normal view reference so that every face image can transform to that pose.

However, one can only identify the frontal normal view qualitatively; it is difficult to pick the absolute normal view out. In this section, we would like to analyze the sensitivity in the performance upon different selection of references. Two tests have been carried out. First we choose consecutive frames in frontal normal view. Since the change in pose among consecutive frames is small, we would expect the alignment should be nearly the same. We verify this in both qualitative and quantitative tests. In our second experiment, instead of choosing consecutive frames, we select different references from the small set of RASL-aligned images where the images are expected to

be well-aligned. We discover that even if we choose the small set of images before RASL, without knowing if it is in frontal normal view, we can still get an acceptable alignment output.

The experiment data applied in our test is the same as that in Section 5.5.2. 5000 consecutive frames with dimension 280×270 were chosen to be the sample. For the small set of RASL-aligned images, we select through uniform sampling: we pick an image out from every 50 image to form a small image set with 100 frames. Their original head poses are not considered in the selection. Since we have found that less batches ensure better alignment, 25-batch Multi-RASL is used in our analysis.

5.6.1 References from consecutive frames

We choose three consecutive frames with frontal normal view as references for our sensitivity analysis. They are shown in Figure 5.9. Although there is noticeable mouth movement among the three frames, the head movement remains small. The qualitative comparison is presented in Figure 5.10. 100 images out of 5000 were chosen for the comparison. It is found that the alignments using different references are almost the same. We further present the quantitative result using average errors of eyebrow corner positions in Table 5.5. The difference is very small, showing the alignment is insensitive to the choice of different frontal normal view references.

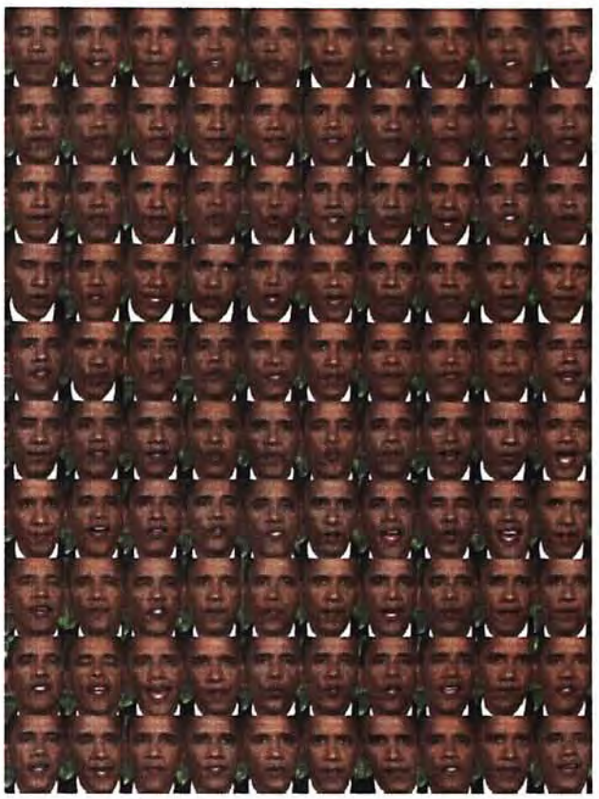
(a) 518th frame (b) 519th frame (c) 520th frame

Figure 5.9: Three consecutive refereneces for our sensitivity analysis

	Left eyebrow	Right eyebrow	Average
518th frame	0.9067	1.9103	1.4085
519th frame	0.8748	1.8957	1.3853
520th frame	0.9833	1.8872	1.4353

Table 5.5: Mean error of eyebrow corner positions of 25-batch Multi-RASL using three consecutive references. Here the distances are measured from the estimated eyebrow corners to their center.

(a) Original images



(b) Using the 518th frame as reference

(c) Using the 519th frame as reference

(d) Using the 520th frame as reference

Figure 5.10: Alignment results using three consecutive references

5.6.2 References from RASL-aligned images

Since it is not always easy to find faces in frontal normal view from the original video sequence, we propose a method to handle this problem. Rather than directly choose the reference from the sequence, we first select a small set of frames from the video and then apply RASL to align them. The resulting images are expected to be aligned well. Empirical results show that we can get an approximate frontal view from RASL provided that the video is taken in frontal view of the person.

We select three images from the small set of 100 RASL-aligned images, as shown in Figure 5.11, to be the reference. Their corresponding original images before alignment are shown in Figure 5.12. The qualitative result is presented in Figure 5.13. It is found that the alignment is acceptable, although there are some small differences among the alignment using different references. For comparison, the alignment results using the original image without aligning with RASL are provided in Figure 5.14 with references shown in Figure 5.12. In Figure 5.14, although the images are aligned to the reference, they are not in frontal normal pose. It is found that our suggestion can achieve a better frontal normal view alignment than random selection from the original images. The eyebrow corner test results are shown in Table 5.6. It is discovered that although the reference images are expected to be aligned to the same pose, the errors in the eyebrow corner positions can still be very distinct. One of the reasons is that the small set RASL is not accurate enough and errors accumulated after two alignment processes. However, the

alignment results are still acceptable for the case that we cannot choose a reference in the frontal normal view directly from the original video sequence.

(a) 51th frame (b) 101th frame (c) 401th frame

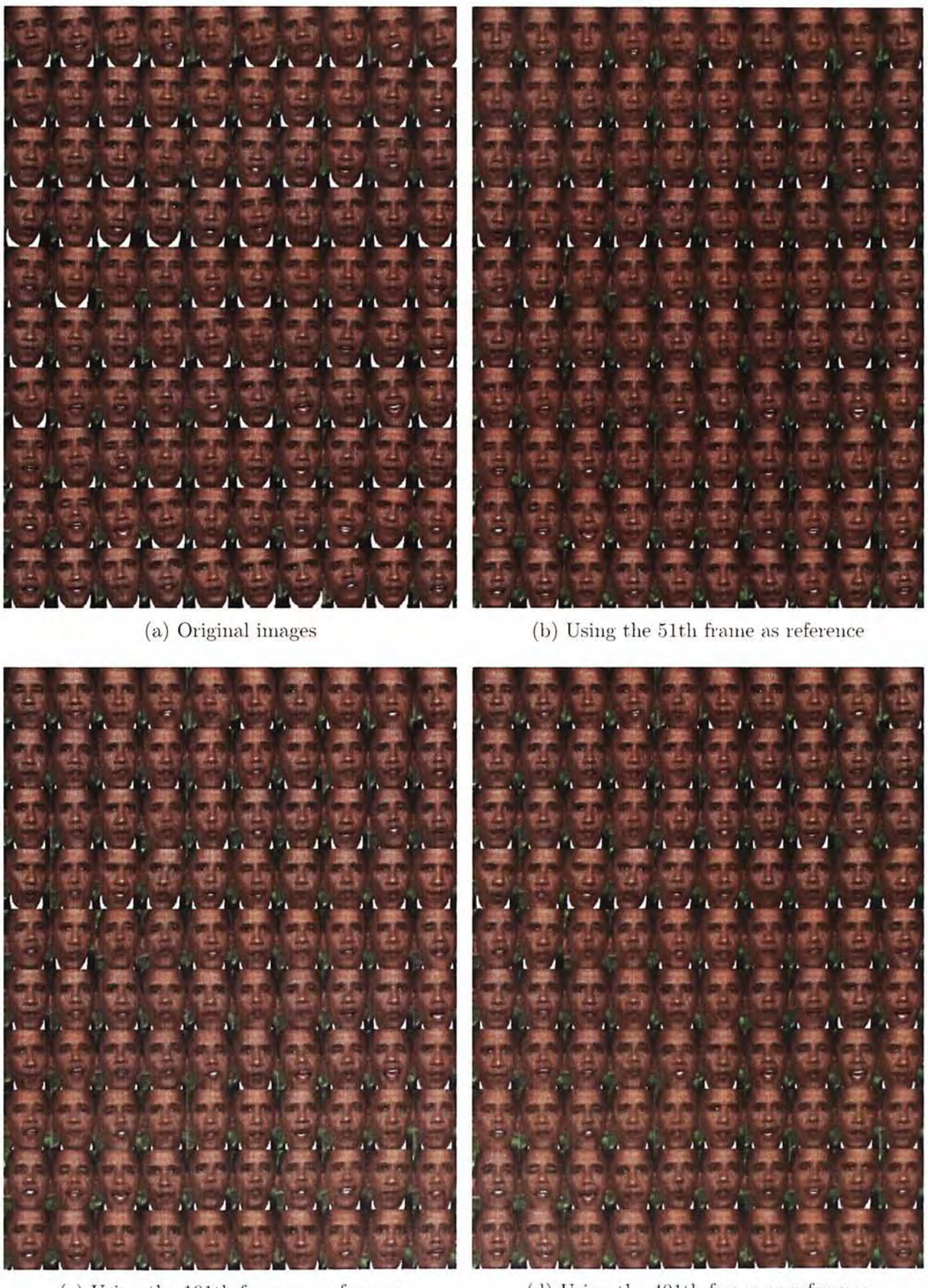
Figure 5.11: Three references chosen from 100 RASL-aligned images

(a) 51th frame (b) 101th frame (c) 401th frame

Figure 5.12: The three chosen reference images before alignment using RASL

5.7 Summary

We have provided a complete explanation on how to apply sparse and low-rank decomposition algorithm for large-scale face alignment problem. Our algorithm is based on the latest advancement of RASL. We extend RASL to Mulit-RASL in order to



(a) Original images

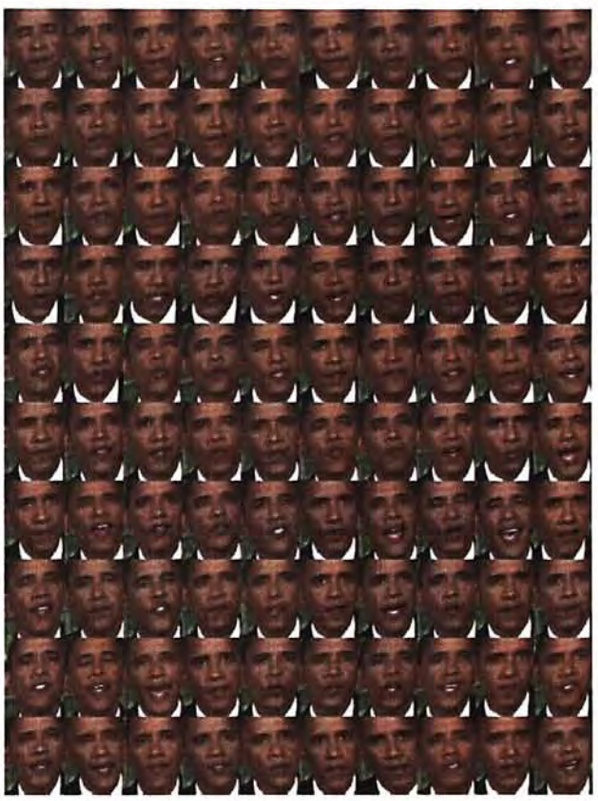
(b) Using the 51th frame as reference

(c) Using the 101th frame as reference

(d) Using the 401th frame as reference

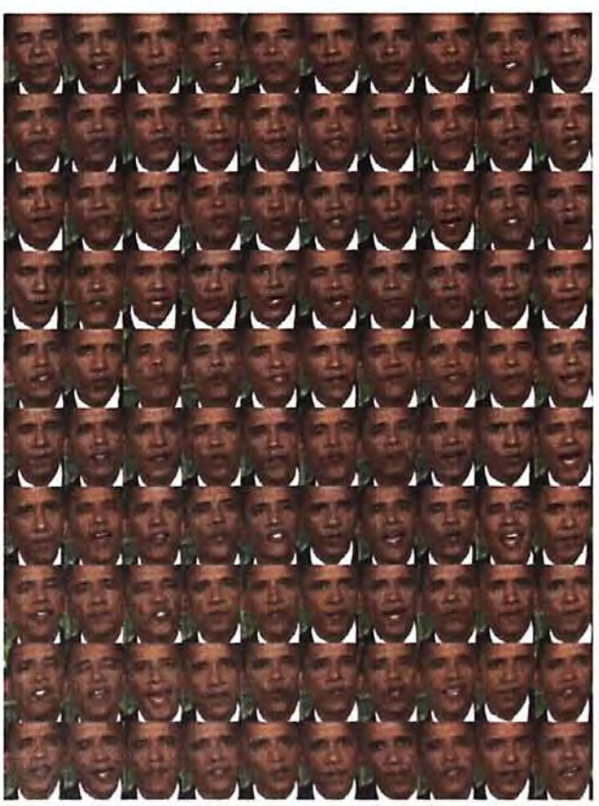
Figure 5.13: Alignment results using three different references from 100 RASL-aligned images

(a) Original images



(b) Using the 51th frame as reference

(c) Using the 101th frame as reference



(d) Using the 401th frame as reference

Figure 5.14: Alignment results using three different reference without alignment with RASL

	Left eyebrow	Right eyebrow	Average
51th frame	1.9555	2.3331	2.1443
101th frame	2.169	3.6629	2.9160
401th frame	1.6941	2.0321	1.8631

Table 5.6: Mean error of eyebrow corner positions of 25-batch Multi-RASL using three references from 100 RASL-aligned images. Here the distances are measured from the estimated eyebrow corners to their center.

tackle the large-scale problem. Through Matlab simulation, it is discovered that our proposed extension not only enables us to apply on large datasets, but also improves the speed and at the same time ensure the accuracy. In addition, we present the comparison of different Multi-RASL approach, using different number of batches. In order to have more accurate alignment, we should use fewer batches. Unfortunately in most situations, the number of batches is lower-bounded by the machine. Thus the misalignment among the batches is impossible to solve using Multi-RASL. Moreover, Multi-RASL is not capable in real-time videos. To deal with these limitations, another sparse and low-rank algorithm is introduced. Details will be explained in the coming chapter.

Chapter 6

Extension of RASL for video: One-by-One Approach

6.1 One-by-One Approach

The main theme of this chapter is to propose an extension of RASL for video, using the one-by-one alignment algorithm. We suggest three choices of l_1-based algorithm which are previously introduced in Chapter 3, including the l_1-regulaized least squares, the l_1-error, and the l_1-regularized l_1-error. Through several experiments we verify the efficacy of our method. It attains comparable quality to RASL while allowing fast and large-scale alignment. The application of our alignment algorithm in photo-real talking head is published in a conference paper [47].

This section gives the motivation of our one-by-one approach. The theory and the algorithm will also be introduced.

6.1.1 Motivation

In Chapter 5, RASL and Mulit-RASL are introduced for face alignment in batch. However, batch alignment is not an efficient method of video sequence since it is not able to exploit the property of video for enhancing its efficiency. Also it is impossible to directly extend RASL to the real-time case. Therefore, we propose the one-by-one approach, which seeks a way to fully utilize the power of sparse and low-rank alignment approach so that it may be possible to tackle the real-time problem.

Besides, our one-by-one approach provides a flexible alternative for alignment. Since the memory required for every image remains unchanged, even in a very limited resources situation such as low memory, our algorithm can still be applied.

6.1.2 Algorithm

The basic principle is to align the $(n+1)$th image using n well-aligned images. Such aligned images can be viewed as a training set or basis, which can be prepared by using RASL. We will show that in our experiment that it is sufficient to use a very small n to align total N face images, i.e. $n << N$.

Recall that the solution of RASL there is a low rank matrix A^*. The first step of our approach is to select n images from the video sequence and align using RASL, producing a low rank solution A^*. Then we use it as a dictionary which acts as the basis to allow forming linear combination to approximate our $(n+1)$th image so that it is aligned with the n RASL-aligned images. Finally, we repeat this step to all the images in the

dataset, regardless of the size of dataset.

Align n images with RASL

In this step, the procedure is basically the same as described in Chapter 5.2. Applying RASL on the n images chosen from the dataset would give us the optimal solutions τ^*, A^*, E^*. We form $\tilde{A} \in \mathbb{R}^{m \times rank(A^*)}$ whose columns consist of $rank(A^*)$ (out of n) independent columns of A^*. It acts as a dictionary for the aligned images, which will be used in the next step. RASL extracts and stores all the important features of the aligned faces in A^*. Therefore, we can use this set of occlusion-free images to be the basis. We would like our dictionary to cover as many features and variations as possible.

Empirical results show that uniformly-spaced selection rather than consecutive sampling can ensure the convergence of τ_{n+1} with fewer selected images. It is reasonable since consecutive frames usually have similar features (e.g. the variations in illumination), which do not provide sufficient variations to form the basis.

In choosing the n input images, there is a tradeoff between quality of dictionary and computational cost. Selecting more images (larger n) would probably lead to a better dictionary, however at the same time increases the speed of computation, mainly due to increase in size of dictionary.

From n to $n+1$

Here we are going to align an additional image with those n images already aligned by RASL. The principle of our method is to search for an optimal transformation parameter of the $(n+1)$th image τ_{n+1} such that $\tilde{A}x$ forms the best approximation of $I_{n+1} \circ \tau_{n+1}$. However, there can be different criteria for the quality of the approximation. We propose three different criteria which can be formulated into optimization problems. They are the l_1-regulaized least squares, the l_1-error, and the l_1-regularized l_1-error respectively. The details will be discussed in Section 6.2.

This step can be divided into **two parts**: outer loop and inner loop. The outer loop is the process of iterative linearization (shown in Algorithm 6.1). Inside the loop, there is an inner loop for the algorithm mentioned in the previous paragraph with split Bregman method which is fast and efficient. As we will see in Section 6.3, if the initial misalignment is not too large, this iteration recovers the correct transformations τ_{n+1} in an efficient manner.

From n to N

We apply the above step to all the remaining images. The $\tilde{A}$ is kept unchanged. Therefore, the memory usage is independent of the number of images N in the dataset. Empirically, we obtain comparable results to RASL in a reasonable time for thousands of images.

Algorithm 6.1 (Outer loop)

INPUT: Image $I_{n+1} \in \mathbb{R}^{w \times h}$, RASL solution A^*, initial transformation τ_{n+1}^0 in affine group, weight μ

WHILE not converged **DO**

Step 1: compute Jacobian matrices w.r.t. transformation:

$$J \longleftarrow \frac{\partial}{\partial \zeta} \left(\frac{vec(I_{n+1} \circ \zeta)}{\|I_{n+1} \circ \zeta\|_2} \right) \Big|_{\zeta = \tau_{n+1}}$$

Step 2: warp and normalize the images:

$$I_{n+1} \circ \tau_{n+1} \longleftarrow \frac{vec(I_{n+1} \circ \tau_{n+1})}{\|vec(I_{n+1} \circ \tau_{n+1})\|_2}$$

Step 3 (inner loop): solve the optimization so as to search for an optimal transformation parameter of the $(n+1)$th image τ_{n+1} such that $\tilde{A}x$ forms the best approximation of $I_{n+1} \circ \tau_{n+1}$

Step 4: update transformation:

$$\tau_{n+1} \longleftarrow \tau_{n+1} + \Delta\tau_{n+1}^*$$

END WHILE

OUTPUT: solution τ_{n+1}

6.2 Choices of Optimization

We introduce three choices of optimization for the inner loop in Algorithm 6.1. The algorithms follow the split Bregman approach described in Chapter 4. As we will see, the computation is fast so it allows efficient alignment.

6.2.1 l_1-Regularized Least Squares

First, we formulate the inner loop in Algorithm 6.1 using the following l_1-regularized least squares (L1LS) problem:

$$\min_{x,\tau_{n+1}} \frac{1}{2} \left\| I_{n+1} \circ \tau_{n+1} - \tilde{A}x \right\|_2^2 + \mu \|x\|_1 \tag{6.1}$$

Here $\tilde{A}$ is the dictionary we defined in Section 6.1. The goal of this optimization is to search for optimal τ_{n+1} such that $\tilde{A}x$ forms the best approximation of $I_{n+1} \circ \tau_{n+1}$ with the least number of columns of $\tilde{A}$. x is a vector with dimension $rank(A^*)$ which represents the coefficients of the linear combination by columns of $\tilde{A}x$. μ is the weight that trades off the least square error and the sparsity of x.

However, the above optimization 6.1 is non-linear which is hard to solve. Similar to that in RASL, we manage to linearize the optimization with iterative linearization. We write $I_{n+1} \circ \tau_{n+1} = I_{n+1} \circ (\tau_{n+1}^0 + \Delta\tau_{n+1}) \approx I_{n+1} \circ \tau_{n+1}^0 + J\Delta\tau_{n+1}$, where J is the Jacobian matrix with respect to the affine transform τ_{n+1}. Thus, the minimization (6.1) becomes

$$\min_{x,\Delta\tau_{n+1}} \frac{1}{2} \left\| I_{n+1} \circ \tau_{n+1}^0 + J\Delta\tau_{n+1} - \tilde{A}x \right\|_2^2 + \mu \|x\|_1 \tag{6.2}$$

which can easily be rewritten into the usual form of l_1-LS:

$$\min_y \frac{1}{2} \left\| I_{n+1}^0 - By \right\|_2^2 + \mu \left\| Cy \right\|_1 \tag{6.3}$$

where $I_{n+1}^0 = I_{n+1} \circ \tau_{n+1}^0$, $B = \begin{bmatrix} \tilde{A} & -J \end{bmatrix}$, $y = \begin{bmatrix} x & \Delta\tau_{n+1} \end{bmatrix}^T$, and $C = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}$. (6.2) can be solved by L1LS in Algorithm 4.2.

Since the linearization only holds locally, in order to find the minimal solution of (6.1), we have to repeat (6.2) about our current estimation of τ_{n+1}^0 for many times until it converges.

Empirically, we find that l_1-LS is more stable than the conventional least squares (LS). LS has similar performance in the case when only Gaussian noise exists in the image I_{n+1}. However, even a small amount of non-Gaussian noise would generate many extra local minima in the objective function, leading to an incorrect optimal solution. The additional l_1-regularized term can act as a smoother, which eliminates the unwanted local minima by penalizing the number of atoms used in the dictionary $\tilde{A}$ to form the approximation.

6.2.2 l_1-Error

Next, rather than using L1LS, we replace the objective function in (6.1) by an l_1-norm minimization:

$$\min_{x,\tau_{n+1}} \left\| I_{n+1} \circ \tau_{n+1} - \tilde{A}x \right\|_1 \tag{6.4}$$

Again, using similar trick as in last subsection, we linearize the optimization by letting $I_{n+1} \circ \tau_{n+1} = I_{n+1} \circ (\tau_{n+1}^0 + \Delta\tau_{n+1}) \approx$

$I_{n+1} \circ \tau^0_{n+1} + J\Delta\tau_{n+1}$, where J is the Jacobian matrix with respect to the affine transform τ_{n+1}. Thus, (6.4) becomes

$$\min_{x,\Delta\tau_{n+1}} \left\| I_{n+1} \circ \tau^0_{n+1} + J\Delta\tau_{n+1} - \tilde{A}x \right\|_1 \tag{6.5}$$

It can be easily converted into the l_1-error form as (4.40):

$$\min_{y} \|e\|_1 \ \ s.t.\ e = I^0_{n+1} - By \tag{6.6}$$

where $I^0_{n+1} = I_{n+1} \circ \tau^0_{n+1}$, $B = \begin{bmatrix} \tilde{A} & -J \end{bmatrix}$, $y = \begin{bmatrix} x & \Delta\tau_{n+1} \end{bmatrix}^T$.

As discussed in Chapter 3 Section 3.2, l_1-error optimization is nearly identical to minimizing the sparse corruption in the test image. In our video of face, the mouth is continuously moving and the eyes are frequently blinking. Since the eyes and mouth only covers a relatively small area of the face, and they are supposed to be at the same position on the face, it is reasonable to consider the movement of the eyes and mouth as the sparse corruption in the images. This idea is consistent with that of RASL, where the sparse matrix S representing the sparse difference between the face for align and the low-rank face.

6.2.3 l_1-Regularized l_1-Error

Now we combine the advantages of l_1-regularization and l_1-error to formulate the l_1-regularized l_1-error for solving our alignment problem as follows:

$$\min_{x,\tau_{n+1}} \left\| I_{n+1} \circ \tau_{n+1} - \tilde{A}x \right\|_1 + \mu \|x\|_1 \tag{6.7}$$

Using the linearization trick, let $I_{n+1} \circ \tau_{n+1} = I_{n+1} \circ (\tau^0_{n+1} + \Delta\tau_{n+1}) \approx I_{n+1} \circ \tau^0_{n+1} + J\Delta\tau_{n+1}$, where J is the Jacobian matrix with respect to the affine transform τ_{n+1}. (6.7) becomes a

linearized optimization problem:

$$\min_{x,\Delta\tau_{n+1}} \left\| I_{n+1} \circ \tau_{n+1}^0 + J\Delta\tau_{n+1} - \tilde{A}x \right\|_1 + \mu \|x\|_1 \tag{6.8}$$

By transforming it into the usual L1L1 form:

$$\min_y \left\| I_{n+1}^0 - By \right\|_1 + \mu \|Cy\|_1 \tag{6.9}$$

where $I_{n+1}^0 = I_{n+1} \circ \tau_{n+1}^0$, $B = \begin{bmatrix} \tilde{A} & -J \end{bmatrix}$, $y = \begin{bmatrix} x & \Delta\tau_{n+1} \end{bmatrix}^T$.

6.3 Experiments

In this section we use the 5000-images dataset of Obama's speech in Section 5.5.2 again to perform the test of our one-by-one approach. The experiments are carried out in Matlab. The stopping criterion of outer loop is when $\|\Delta\tau_{n+1}\|_2 \leq 10^{-6}$ or number of iterations exceeds 300. For L1LS, the weight μ in 6.1 is set to be 10^{-3}, while for L1L1, $\mu = 10^{-2}$. We choose $n = 100$ samples uniformly to form our dictionary by performing RASL to align them. The initial transformation is taken to be identity for all frames.

6.3.1 Evaluation for Different l_1 Algorithms

We verify the accuracy of our algorithm using both quantitative and qualitative evaluation. Similar to the experiment of RASL and Multi-RASL in Chapter 5, we identify the eyebrow corners positions as fair features for quantitative comparisons of our proposed algorithms.

Quantitative Evaluation

It is found that not all the frames achieve convergence within 300 iterations. This may due to the original orientation of the head in some frames is too extreme. The number of frames that unable to converge in 300 iterations is given in Table 6.1. In fact, the convergence depends on the initial condition as well as the complexity of the objective function. From Table 6.1 we discover that L1LS has the largest number of non-convergent frames, while L1L1 has the least. This may reveal that the objective function of L1LS is not as good as L1L1 for face alignment.

Algorithm	non-convergent frames
L1LS	23
l_1 error	11
L1L1	8

Table 6.1: Number of frames that are not converge within 300 iterations

Next the eyebrow corners test is considered. Same as in previous chapter, in order to eliminate the effect of scaling effect on our quantitative comparison, we normalize the distance between the left and right eyebrow corners to be 50 pixels. We neglect the non-convergent frames. In last chapter, we have already given the statistics of errors for the 100 images dictionary in Table 5.3. This time we do not have one specific reference, therefore we use the mean eyebrow corners positions of the 100 images in dictionary as the reference for comparison. The statistics of errors in eyebrow corners are as shown in Table 6.2. It is found that the errors are approximately the same as that of RASL results for the 100 images, and smaller than that of Multi-RASL

(comparing with Table 5.2). The alignment results for the three different algorithms are very similar in this case.

		Left eyebrow	Right eyebrow	Average
Mean error	(a)	1.5176	1.8746	1.6961
	(b)	1.5078	1.8700	1.6889
	(c)	1.5030	1.8757	1.6894
Standard error	(a)	0.7453	0.8367	0.7910
	(b)	0.7401	0.8297	0.7849
	(c)	0.7377	0.8305	0.7841
Maximum error	(a)	4.5665	4.3926	4.4800
	(b)	4.5678	4.3938	4.4808
	(c)	4.5666	4.3927	4.4797

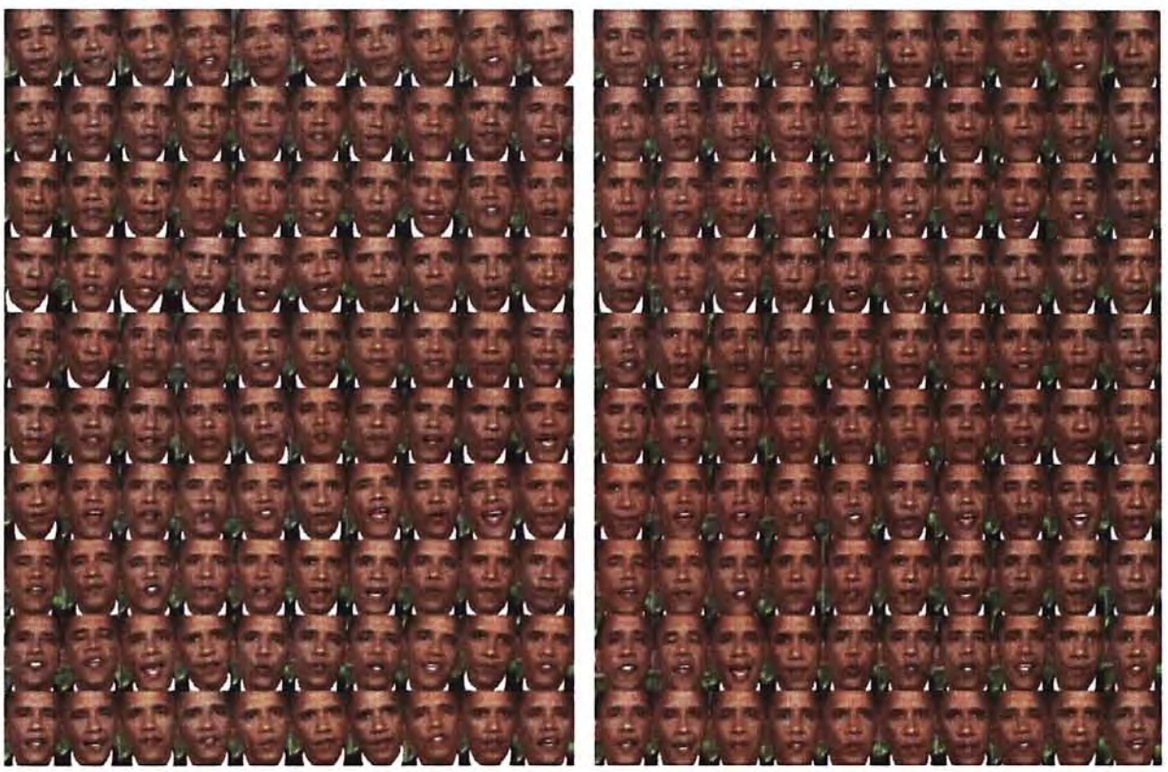
Table 6.2: Eyebrow corners comparison with (a) L1LS, (b) l_1-error and (c) L1L1. Here the distances are measured from the estimated eyebrow corners on faces to mean eyebrow corners positions.

Qualitative Evaluation

To illustrate the effectiveness of our one-by-one approach, we provide also the alignment outputs using the three l_1-algorithms for qualitative comparisons in Figure 6.1. The eyebrow corners are marked in white as reference.

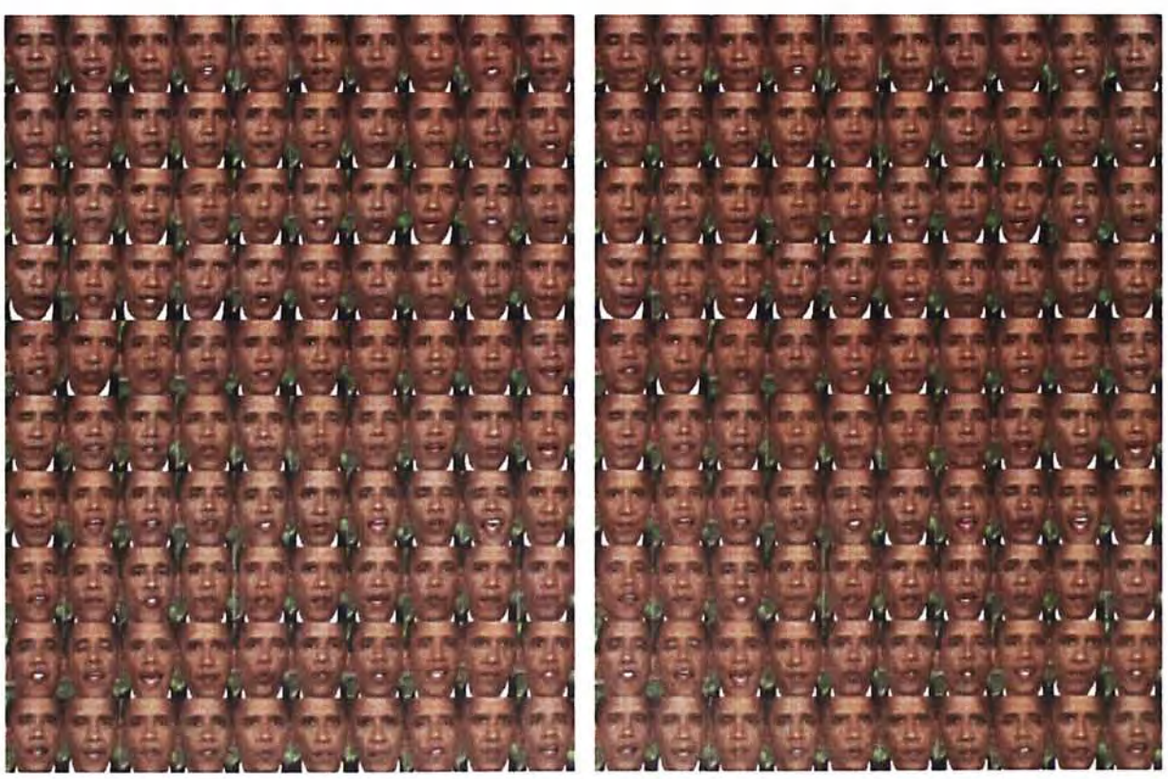
Memory and Speed

The problem of limited memory always exists due to physical constraints. RASL requires to store all N images in memory during alignment. It may be practical for hundreds of images, but not for thousands or millions of images. In comparison, our algorithm only have to store $n + 1$ frames in memory at a



(a) Original images

(b) L1LS



(c) l_1-error

(d) L1L1

Figure 6.1: 100 uniformly sampled images (with eyebrow corners marked) to illustrate efficiency of our one-by-one approach

time, and n can be flexible depending on the dataset and the maximum memory size.

The speed of RASL depends much on the number of outer loop iterations as the inner loop algorithm for sparse and low rank decomposition is extremely slow (because of Singular Value Decomposition). For images with large misalignments, the time required would increase significantly as more outer loop iterations are needed. In contrast, we employ fast split Bregman algorithm in the inner loop of our algorithm. Assume we are given a dictionary with all necessary features captured within, then the number of outer iterations are similar for each images, thus the overall computational time is linearly proportional to the number of images N in the dataset. In above experiment, the approximate computation time for different l_1-algorithms is as shown in Table 6.3. It is found that l_1-error is the fastest among the three. Unfortunately, it is still far from real-time speed.

Algorithm	Approximate Time required
L1LS	34720s
l_1-error	26980s
L1L1	38530s

Table 6.3: Approximate running time for 5000 images in one-by-one experiment using different l_1-algorithms

6.3.2 Conclusion

Comparing with Multi-RASL, one-by-one approach attains a better alignment performance for most of the images. However,

the average time per frame is longer than that of Multi-RASL. This may be caused by those non-convergent frames. Actually, 90% of the frames can be aligned within 50 iterations. Thus the running time can be much faster if the maximum number of iterations is limited to a smaller number, for example, 100 iterations. However, this risks an increase in number of non-convergent frames which may lead to misalignment. Although our one-by-one approach is not fast enough to achieve real-time speed, it lays the foundation for efficient face alignment for video sequence. The improvement in processing power of computers nowadays would definitely allow real-time processing in future.

Among the three l_1-algorithms, L1L1 gives the best performance in terms of convergence. In theory, L1LS is not as robust to occlusion as l_1-error and L1L1. In some cases where the eyes are blinking, there will be an elongation effect on faces. Our experiment also shows that L1LS it is not as efficient as l_1-error and L1L1. Therefore, for efficient and accurate face alignment, the l_1-error based algorithms, including L1L1, are recommended.

6.4 Exploiting Property of Video

In this section, we demonstrate the use of video property to enhance our proposed algorithm. In a video sequence, there is usually a relation between consecutive frames. One relation in the face video of our application is that the movement of the head is slow and continuous. Therefore, the transformation change is expected to be small between the frames. This leads to a way to increase the speed of convergence and eventually

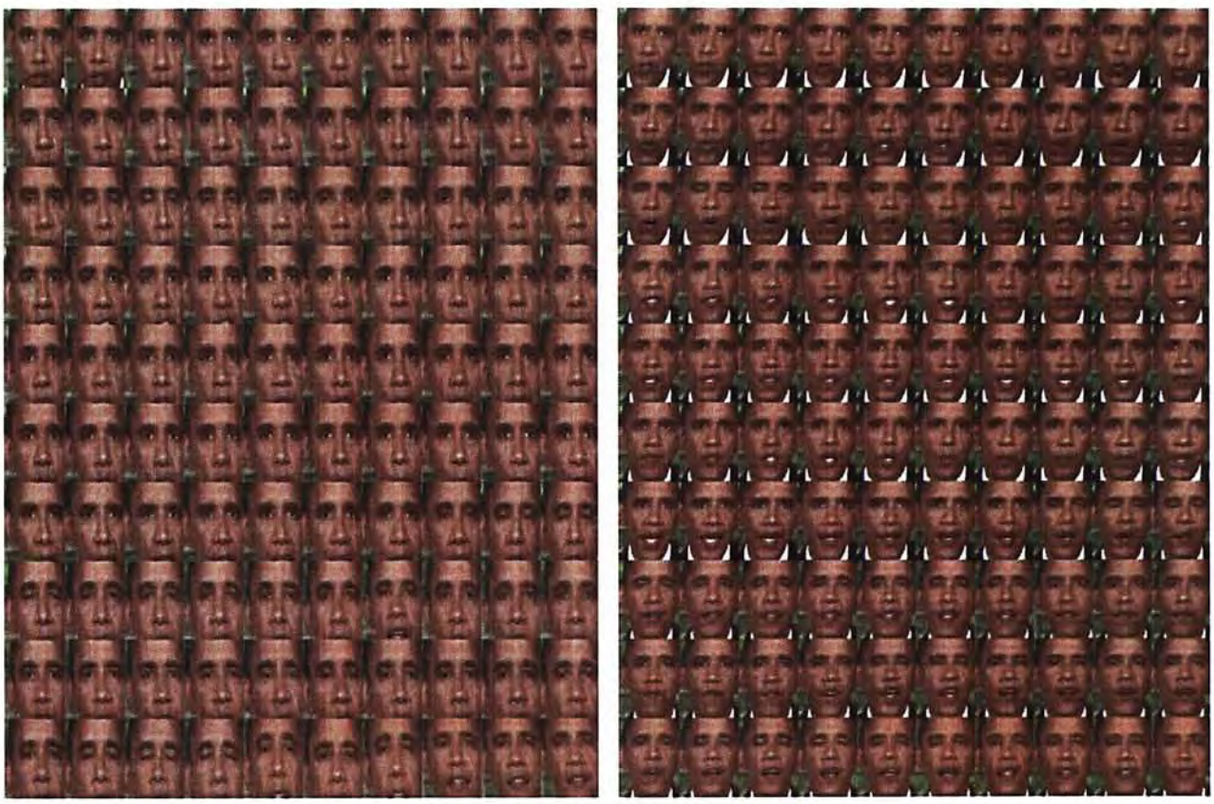
lower the overall computational cost.

Assume the previous frame is aligned appropriately, we use that transformation parameters τ to be the initial transformation parameters for the following frame since the change in transformation parameters is expected to be small. This strategy is verified using the same example as in Section 6.3. The running time for each frame is about 2 seconds, which is aproximately two times faster than using a fixed initial condition.

Unfortunately, this strategy has its disadvantage. If the previous frame has not aligned properly, the next frame would tend to misalignment. Figure 6.2 shows the case where such misalignment occurs. This shows the solution of the optimization is sensitive to the initial condition. One way to prevent it to happen is to use the first initial transformation after perform a specific number of images, say every 50 images, to ensure the error would not be inherited by successive frames.

6.5 Summary

The one-by-one l_1-based face alignment approach is proposed. The details are fully illustrated and explained. Three algorithms are tested, including L1LS, l_1-error and L1L1. Both qualitative and quantitative evaluation shows that one-by-one approach outperforms batch Multi-RASL approach in Chapter 5. Moreover, among our three l_1-algorithms, L1L1 performs the best. Therefore, for efficient and accurate large-scale face alignment, one-by-one L1L1 is recommended.



(a) Initial transform parameters using final transform of previous frame

(b) Using a fixed initial transform parameters

Figure 6.2: Alignment error occurs in our strategy of exploiting video property

Chapter 7

Conclusion and Future Work

In this thesis, the efficient and accurate face alignment approach is introduced, which is based on the recent advancement in sparse and low-rank optimization theory. The latest development of sparse representations is reviewed and generalized. Moreover, the sparse representation techniques are extended to deal with sparse corruption problem, where l_1-error is introduced based on l_1-norm minimization. Extensive experiments have shown that our proposed l_1-error approach is promising and applicable in various image processing problems such as face alignment.

In addition, the algorithms for l_1-error and its variants were derived using split Bregman algorithm. It is proved that split Bregman approach is one of the fastest algorithm specialized for l_1-problem. Furthermore, our experiment using surveillance videos illustrates the efficiency of our algorithms.

Aiming at solving the large-scale face alignment problem, two techniques have been proposed. The first one is named Multi-RASL, which can be viewed as a direct extension of RASL.

Original RASL is not capable in the large-scale case where the memory restrains its ability. Since RASL has been proven to achieve high precision alignment of images both in theory and in practice, our extension also ensures a reliable result. The trick of Multi-RASL is to set a reference and divide the whole dataset into smaller batches. For each batch, we align all the images within to the reference pose based on the ordinary RASL. By performing this process for all batches, all the face images can be aligned to the reference in theory.

Although Multi-RASL provides a reliable result, it still has some weaknesses. First, if the number of batches increases, the misalignment among different batches becomes significant. Another disadvantage of Multi-RASL is that if a few more new images are added to the dataset, the whole computation may have to start over again in order to prevent large misalignment for the newly added images. In contrast to the batch alignment in RASL, we introduce the one-by-one approach, which tries to prevent the weaknesses of Multi-RASL. We modify the tricks of convex relaxation and iterative linearization in RASL to satisfy our requirement using l_1-norm minimization algorithms including L1LS, l_1-error and L1L1. Experiments have discovered that our one-by-one approach outperforms the batch Multi-RASL. Besides, L1L1 has shown the best performance in handling our face alignment task.

In fact, our preliminary result of using one-by-one L1LS has been published in [47]. That work is collaborated with the speech group in Microsoft Research Asia (MSRA). There is a project of talking head in which accurate face alignment is a

crucial step, since even a small misalignment would cause the generated talking head unnatural. We have tested out our algorithm using real data for the talking head. It improves over the traditional method (3D pose tracking). However, due to the copyright issue, the results cannot be completely presented here. The investigation of better face alignment for the talking head is still on-going.

Here we list several possible directions of work in future based on our contribution, both in face alignment and also sparse and low-rank approach:

1. We have developed the framework for efficient and accurate face alignment, aimed at improving the conventional talking head technology. During the alignment, we obtain the affine transform parameters for each image. A question is raised: can we generalize a relation between the head movement with the transform parameters? The parameters are somehow representing the 3-D orientations of the head. If we are able to find out the relationship in between, it is possible to generate natural head movement using the transformation parameters. That would lead to a huge improvement in talking head technology.

2. The efficiency of our one-by-one approach depends greatly on the speed of l_1-algorithm. The further development of faster l_1-algorithms may allow more efficient alignment and even achieve the real-time speed. Besides, our face alignment approach can be set as one of the benchmark for the l_1-norm minimization algorithms.

3. We are dealing with video clips of faces, which is very common in nowadays technology such as instant messenger and video conferencing. The possibility of transplanting our algorithm to these applications is worth investigating. Besides, it may be able to combine with the technology of cell phones and develops apps for both entertainment and educational purposes.

4. As mentioned in Chapter 3, there is some similarities between our proposed l_1-error minimization with the least absolute deviation (LAD) technique in regression. LAD has been an old discipline when fast algorithm is not yet developed. Our derived fast algorithm of l_1-error using split Bregman trick may be able to improve the traditional LAD applications in speed and accuracy. Further investigations have to be conducted.

5. We have generalized the essence of sparse and low-rank techniques and we have also seen the capability of such techniques in image processing. One possible direct application is in video segmentation. For example, we get a video clip from a movie, how to automatically identify and segment into different scene, and for each scene, is it possible to obtain a key frame to conclude that scene? In our experiment using surveillance videos, we have seen that L1L1 has the property to identify the correspondence of selected frame to which video. Also in one scene, the background remains unchanged while the camera may involve in a motion such as translation and rotation, as well as zooming in and out.

In fact, this is similar to our face alignment problem where the face may be translated, rotated and scaled up or down. Therefore, I believe similar techniques may be able to apply on video segmentation problems.

Appendix A

Appendix

In this appendix, the derivation of solution for the simplest form of l_1-regularized least squares problem is presented.

Theorem A.1. *Solution of the l_1-regularized least squares:*

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|x-u\|_2^2 \tag{A.1}$$

is given by the shrinkage operator

$$x_i = shrink(u_i, 1/\lambda) \tag{A.2}$$

where x_i and u_i are the i^{th} entries of x and u respectively. The shrinkage operator is defined as $shrink(x,\gamma) = \frac{x}{|x|}\cdot\max(|x|-\gamma, 0)$.

Proof. First we take the first derivative of the objective function with respect to x and set it to be 0:

$$0 = \frac{d}{dx}\left[\|x\|_1 + \frac{\lambda}{2}(x-u)^T(x-u)\right] \tag{A.3}$$

$$= \frac{d}{dx}\|x\|_1 + \lambda(x-u) \tag{A.4}$$

Notice that A.3 is separable, i.e. entries are independent with each other. Let us consider the i^{th} entry,

$$0 = \frac{d}{dx}|x_i| + \lambda(x_i - u_i) \tag{A.5}$$

To deal with the absolute value $|x_i|$, we separate into different cases: When $x_i > 0$,

$$x_i = u_i - \frac{1}{\lambda} > 0 \tag{A.6}$$

When $x_i < 0$,

$$x_i = u_i + \frac{1}{\lambda} < 0 \tag{A.7}$$

Then we obtain the solution of x_i by

$$x_i = \begin{cases} |u_i| - 1/\lambda & for\ u_i > 1/\lambda \\ -|u_i| + 1/\lambda & for\ u_i < -1/\lambda \\ 0 & for\ -1/\lambda < u_i < 1/\lambda \end{cases} \tag{A.8}$$

Finally, A.8 can be generalized using the shrinkage operator

$$x_i = shrink(u_i, 1/\lambda) \tag{A.9}$$

□

Bibliography

[1] P. Bloomfield and W. Steiger. Least absolute deviations curve-fitting. *SIAM Journal on Scientific and Statistical Computing*, 1(2):290–301, 1980.

[2] P. Bloomfield and W. Steiger. *Least absolute deviations: theory, applications, and algorithms*. Birkhcauser, 1983.

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[4] L. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[5] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[6] E. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Arxiv preprint arXiv:0912.3599*, 2009.

[7] E. Candes and J. Romberg. l1-magic: Recovery of sparse signals via convex programming. *URL: www.acm.caltech.edu/l1magic/downloads/l1magic.pdf*.

[8] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[9] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(6):681–685, 2001.

[10] T. Cootes, C. Taylor, D. Cooper, J. Graham, et al. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[11] E. Cosatto and H. Graf. Photo-realistic talking-heads from image samples. *Multimedia, IEEE Transactions on*, 2(3):152–163, 2002.

[12] M. Cox, S. Sridharan, S. Lucey, and J. Cohn. Least squares congealing for unsupervised alignment of images. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.

[13] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004.

[14] D. Donoho. For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829, 2006.

[15] D. Donoho and M. Elad. Optimally sparse representation in general (non-orthogonal) dictionaries via L1 minimization. *Proc. Natl. Acad. Sci.*, 100:2197–2202, 2003.

[16] D. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *Information Theory, IEEE Transactions on*, 47(7):2845–2862, 2002.

[17] D. Donoho and P. Stark. Uncertainty principles and signal recovery. *SIAM Journal on Applied Mathematics*, 49(3):906–931, 1989.

[18] D. Donoho and J. Tanner. Neighborliness of randomly projected simplices in high dimensions. *Proceedings of the National Academy of Sciences of the United States of America*, 102(27):9452, 2005.

[19] M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586–597, 2008.

[20] T. Goldstein and S. Osher. The split Bregman method for L1 regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[21] R. Gribonval and M. Nielsen. Sparse decompositions in unions of bases. *IEEE Trans. Inform. Theory*, 49(12):3320–3325, 2003.

[22] L. Gu and T. Kanade. 3d alignment of face in a single image. In *Computer Vision and Pattern Recognition, 2006*

IEEE Computer Society Conference on, volume 1, pages 1305–1312. IEEE, 2006.

[23] G. B. Huang, V. Jain, and E. Learned-Miller. Unsupervised joint alignment of complex images. *Computer Vision, IEEE International Conference on*, 0:1–8, 2007.

[24] W. Kienzle, G. Bakir, M. Franz, and B. Scholkopf. Face detection-efficient and rank deficient. *Advances in Neural Information Processing Systems*, 17:673–680, 2005.

[25] D. Kim, J. Kim, S. Cho, Y. Jang, S. Chung, and B. Kim. Progressive AAM based robust face alignment. In *Proc. of world academy of science, engineering and technology*, volume 21, pages 488–492. Citeseer, 2007.

[26] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. A method for large-scale l1-regularized least squares problems with applications in signal processing and statistics. *IEEE Journal on Selected Topics in Signal Process*, 2007.

[27] S. Z. Li, H. Zhang, Y. ShuiCheng, and Q. Cheng. Multi-view face alignment using direct appearance models. *Automatic Face and Gesture Recognition, IEEE International Conference on*, 0:0324, 2002.

[28] Z. Lin, M. Chen, L. Wu, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *Arxiv preprint arXiv:1009.5055*, 2010.

[29] D. Malioutov, M. Cetin, and A. Willsky. Homotopy continuation for sparse signal representation. In *Acoustics, Speech,*

and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on, volume 5. IEEE, 2005.

[30] Y. Nesterov. A method of solving a convex programming problem with convergence rate O (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[31] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling and Simulation*, 4(2):460–489, 2006.

[32] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition*. Citeseer, 2010.

[33] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003.

[34] J. Pluim, J. Maintz, and M. Viergever. Mutual-information-based registration of medical images: a survey. *Medical Imaging, IEEE Transactions on*, 22(8):986–1004, 2003.

[35] D. Pollard. Asymptotics for least absolute deviation regression estimators. *Econometric Theory*, 7(02):186–199, 1991.

[36] S. Shan, Y. Chang, W. Gao, B. Cao, and P. Yang. Curse of mis-alignment in face recognition: Problem and a novel mis-alignment learning solution. *Automatic Face and Ges-*

ture Recognition, IEEE International Conference on, 0:314, 2004.

[37] V. Smith and T. Hall. A comparison of maximum likelihood versus blue estimators. *The Review of Economics and Statistics*, 54(2):186–190, 1972.

[38] X. Tai and C. Wu. Augmented Lagrangian method, dual methods and split Bregman iteration for ROF model. *Scale Space and Variational Methods in Computer Vision*, pages 502–513, 2009.

[39] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *Submitted to SIAM Journal on Optimization*, 2008.

[40] A. Vedaldi, G. Guidi, and S. Soatto. Joint data alignment up to (lossy) transformations. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[41] L.-J. Wang, X.-J. Qian, W. Han, and F. Soong. Synthesizing photo-real talking head via trajectory-guided sample selection. In *Proc. Interspeech*, pages 446–449, 2010.

[42] P. Wang, L. Tran, and Q. Ji. Improving face recognition by online image alignment. *Pattern Recognition*, 1:311–314, 2006.

[43] Q. Wang, W. Zhang, X. Tang, and H. Shum. Real-time bayesian 3-d pose tracking. *Circuits and Systems*

for Video Technology, IEEE Transactions on, 16(12):1533–1541, 2006.

[44] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *Submitted to Journal of the ACM*, 2009.

[45] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

[46] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 210–227, 2008.

[47] K. Wu, L. Wang, F. Soong, and Y. Yam. A sparse and low-rank approach to efficient face alignment for photo-real talking head synthesis. *The 36th International Conference on Acoustics, Speech and Signal Processing*, 2011.

[48] A. Yang, A. Ganesh, Z. Zhou, S. Sastry, and Y. Ma. Fast l1-minimization algorithms and an application in robust face recognition: a review. In *Proceedings of the International Conference on Image Processing*, 2010.

[49] J. Yang and Y. Zhang. Alternating direction algorithms for l1-problems in compressive sensing. *Arxiv preprint arXiv:0912.1185*, 2009.

[50] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for l1-minimization with applications to compressed sensing. *SIAM J. Imaging Sci*, 1(1):143–168, 2008.

[51] Z. Zhang, Z. Liu, D. Adler, M. Cohen, E. Hanson, and Y. Shan. Robust and rapid generation of animated faces from video images: A model-based modeling approach. *International Journal of Computer Vision*, 58(2):93–119, 2004.