Image patch modeling in a light field

Zeyu Li

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-81
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-81.html

May 15, 2014

Image patch modeling in a light field

by

Zeyu Li

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ruzena Bajcsy, Chair
Professor Avideh Zakhor
Professor Kannan Ramchandran
Professor Bruno Olshausen
Harlyn Baker

Spring 2014

Image patch modeling in a light field

Abstract

Image patch modeling in a light field

by

Zeyu Li

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ruzena Bajcsy, Chair

Understanding image content is one of the ultimate goals of computer vision, and effectively and efficiently extracting features from images is a key component of all vision research. This thesis discusses methods related to an image-patch based approach to this feature analysis. Image-patch based methods have attracted a lot of interest for the analysis of a single images in application areas such as visual object recognition, image denoising, and super-resolution computation. The basic idea is to treat a single image as a collection of independent image patches, each of which can be encoded by, for example, a sparse coding model. The global characterization of that image is attained by aggregating the patch codes, which brings some level of shift-invariance and robustness to image noise and signal degradation.

In this thesis, a new scheme, *scene geometry-aware image-patch modeling*, based on the concept of a **patch-cube**, is proposed to model image patches in a light field, rather than in a single image. A light field is a collection of images all acquired at the same instant, providing a set of perspectives on the scene as though observing all of the light information that passes through a windowing portal (clearly with some discretization and sampling). The scene geometric information is implicitly incorporated in our modeling process, including depth and occlusion, without explicit knowledge of 3D scene structure. These extra constraints on the scene geometry empower our learned features to be less affected by image noise, lighting conditions, etc. As demonstration, we apply our method to joint image denoising and joint spatial/angular image super-resolution tasks, where its use of the light field will be seen to permit it to outperform its image-patch based counterparts. Here, a 2D camera array with small incremental baselines is used to capture the light field data, and this analysis is the majority of what we report. Additionally, working with real data from real light-field cameras, we present novel and highly effective methods for the calibration of these camera arrays.

In common with the single-image model, learning a good "dictionary" plays a very important role in our work – selecting an appropriate set of features that can provide succinct representations of a scene. Inspired by the success of the image patch-based method [2], we

show that feature extraction for image patches is closely related to the low-rank kernel matrix approximation using the Nystrom method. The dictionary in sparse coding, or cluster centers in K-means clustering, are actually landmark points which can better capture the underlying higher-dimensional (manifold) structure of the data. Based upon this observation, our contribution is two fold: 1) an efficient algorithm to perform Kernel Principle Component Analysis feature extraction using landmark points, and 2) an alternative method for finding better landmark points based on *Generalized Extreme Value distribution*s, GEV-Kmeans.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank to my PhD advisor, Professor Ruzena Bajcsy, for supporting me during the past years. I appreciate all her contributions of time, ideas and patience to help me finishing up my degree at Berkeley. I also have to thank to my dissertation committee members, Professor Avideh Zakhor, Professor Bruno Olshausen, Professor Kannan Ramchandran and Harlyn Baker for their helpful advice and suggestions.

I also have to thank the Berkeley Teleimmersion Lab and the previous and present members, including Gregorij and Ram. I am also very grateful to Irvin Sobel and Harlyn Baker from HP Labs. Especially, Harlyn has been working with me for many years. Without you, I can not get this degree. Thanks Harlyn.

Lastly, I would like to thank my family for being supportive during the course of my PhD study. For my parents who provide unconditional love and care. I especially thank my wife Jing Xue and my son Owen for the support, patience and encouragement.

Chapter 1

Introduction

Over the last few years, considerable effort has been devoted to learning appropriate features for visual data modeling, and this has attracted a lot of interest in the computer vision, psychology and machine learning fields. For a given image, a global representation is obtained by the following layering process [80]: extract image patch features on a regularly-spaced grid over the input image, encode these features (coding step), apply a non-linearity on the codes (non-linearity layer), and aggregate the codes within local neighborhoods in a spatial pyramid fashion (pooling step). Deep learning [33][12][80][76] further proceeds by stacking the above layers together into a multi-layer deep structure, where the upper layers take, as inputs, the outputs from the lower layers. The hope is that simple structuring of the common feature information will make it possible to aggregate the layers in a more systematic way to form a more complex and expressive structure within a deep learning framework.

As a core part, image patch modeling encodes a small local image patch using a statistical model, so that the resulting representation, or code, has some desirable properties such as sparseness, compactness and statistical independence. The bag-of-Words model [63] first generates a codebook by K-means clustering over all training patches, with each patch then mapped to the nearest codeword. Sivic et al. [63] adopt the soft-K-means representation for image patches. Hinton et al. [33] models image patches using a restricted Boltzmann machine. Among many patch models, the sparse coding method [52] encodes each patch using a sparse linear combination of pre-trained dictionary atoms. The dominant features of a local patch, its sharp edges, seem then to be well preserved, which is important to describing the object we are interested in. This type of research has achieved much success in image analysis, for example image denoising [27][11], super-resolution [78][36], image inpainting, visual object recognition [80], etc.

The nature of looking at an isolated image patch from an image introduces some drawbacks. [1] The texture or appearance of an image patch solely can not distinguish a *true*-edge, arising from the color variation across an object from a *false*-edge, caused by other events such as occlusion. The former describes a property of a target object, while the latter is

[1] In an optical flow sense, small image patches introduce an aperture problem.

only indicative of depth discontinuity within the patch. Further the patch appearance or the features derived from it, for example HOG[25], SIFT[43], is affected by image noise, lighting conditions, shifts, distortions and etc. This renders the acquired codes for that patch unstable, susceptible to variation with those changes. In an object recognition task, the pooling layer mentioned above achieves some sort of shift and distortion-invariance by average or maximum pooling over a pre-defined window. In a single image denoising task, denoised image patches are typically averaged within their spatial neighborhoods. Another line of work is to incorporate human-derived priors into the coding stage [32][77]. Grosse et al. [32] learn a shift-invariance feature by constraining the neighboring acoustic features sharing some characteristics, for example a similar pattern of sparseness. However it is in general not very clear what kind of priors are suitable.

3D geometry, on the other hand, is less affected by conditions such as illumination, image noise, relative scale, color variations, etc, and it is felt that this could significantly help the image modeling. Some efforts have exploited this, for example Socher [64] treats depth data as another modality to help image patch modeling. Lai et al. [38] formalizes a new feature by attaching the depth information to the image feature and then uses this concatenated feature to conduct image recognition tasks. Tosic [68] applies the sparse coding model over the depth map in a stereo setting by assuming the depth data have different distributions with respect to distance. All of the above works require the 3D depth information to be dense and accurate. While 3D information definitely aids the image modeling problem, extracting dense depth information, for example from multi-view camera systems or Lidar, still presents a major challenge to the computer vision community.

The light-field camera, on the other hand, is a new framework for image acquisition. It captures images of an object of interest simultaneously from multiple viewpoints, generally in a plane. Light field cameras have some unique advantages in comparison with a single pinhole-imaging camera. For example, light-field cameras (or a 2D camera array that acts as a light-field camera over a larger baseline), by varying the aperture and exposure time of each pin-hole camera lens, provide an effective way to balance the trade-off between depth of field and motion blur. In comparison with the more general unconstrained layout of multiple camera systems, a 2D grid camera array effectively facilitates extracting scene structure. Light-field cameras have been successfully used in computational photography[49], depth recovery [71], image re-focusing[49], and other areas. In comparison with single pinhole cameras, we believe the slight changes in view-point of a light-field camera can provide useful structural information about the object of interest, without requiring accurate 3D depth information, and this also presents more opportunities in image modeling.

In distinction from the micro-sensor 2D grid camera array design, consider our camera grid as shown in Figure 1.1, which presents an effective way to capture a light field. The baseline is around $20mm$. A method for calibrating such systems is presented in Appendix A. The camera grid can be treated as a collection of linear 1D imagery arrays, arranged both horizontally and vertically. By stacking all images in a linear camera array, we formulate a special type of image, termed an Epipolar Planar Image(EPI) [8][16]. The properties of a 1D EPI-image have been well investigated in [8][16], scene properties can be relatively easily

Figure 1.1: 2D camera array

inferred, including depth, depth discontinuities, occlusions, and free space [16]. Although the success of depth recovery from a linear camera array through 1D EPI-image analysis is clear, accurate depth retrieval from a light-field still poses challenges. In order to get sufficient overlap among cameras, the inter-imagery baselines in a 2D camera array must be small. This makes it more difficult to recover depth estimates for objects at a greater distance. In texture-less regions, estimating slope in an EPI will be ill-posed.

In the first part of this dissertation, we will explore the image patch modeling problem by considering image data and scene geometrical information simultaneously and in an integrated way. A geometry-aware sparse coding strategy is then presented to efficiently encode an image patch respecting geometric constraints embedded in a light-field. Specifically, instead of treating each image patch as an isolated object, we consider it as part of a **patch-cube**, which is formalized by stacking the corresponding image patches from either horizontal or vertical viewpoints. The horizontal **patch-cube**, for example, can be also interpreted as a collection of regular image patches from horizontally-aligned viewpoints, EPI images, which implicitly encodes the scene geometry information. This dual view of a **patch-cube** makes it possible to take advantage of the scene geometric information without losing the benefit brought by the sparseness principle.

These above horizontal and vertical **patch-cube** arrangements only consider the *cross*-structure of a 2D camera array, meaning, for an image patch P_{rc}, only the camera on the r-th row and c-th columns are used to provide multiple viewpoints. We further generalize this idea to the full 2D grid of camera. Leveraging the very complete description of a local image patch, both its appearance and the neighbors from 2D viewpoints, leads to more meaningful and robust representations.

We believe our new model to be suitable for different visual tasks. In this dissertation, we apply our method to joint image denoising and joint angular/spatial super-resolution tasks in a light-field. The first application is important due to the fact that different sensors are used in building a 2D camera array. One or more cameras are likely to be degradated or corrupted by noise. Using the remaining "good" images, with the facility of the implicitly embedded geometric information, to recover the noisy one(s) is a more economical solution. Here, we consider the extreme situation where all images are polluted by noise in order to

explore how well the geometric information can help denoising. On the other hand, the second application is of particular importance. Light-field cameras usually have to balance the trade-of between spatial and angular resolution due to the fact that total sensor resolution is limited. The angular resolution is critical to the EPI-based image analysis. The larger the angular resolution, the more viewpoints acquired of the scene, the more accurate the scene geometry that can be extracted through EPI-image analysis. This will not only benefit depth retrieval from a light-field, but also the image analysis based on the EPI-patches. Our goal is to effectively interpolate novel viewpoints over the 2D grids, while still respecting the geometric constraints embedded in a light-field. Performance superior to those of counterpart measures performed on a single image in both applications demonstrates the value of geometric constraints in the image patch modeling problem.

There are few efforts at joint denoising of multiple images or angular/spatial super-resolution in a light field in the literature. Zhang et al. [86] try to denoise more general multi-images, not a 2D camera grid, by assuming knowledge of the correspondences of an image patch across cameras. Wanner et al. [71][72] have attempted to super-resolve a light field. Levin [40] and Mitra [47] model all relevant patches using a Gaussian Mixture Model. All of the above works need pre-computing the 3D depth, and they have not provided a principled way to address the problem.

In the above, our key insight is to treat EPI-patches as a regular image and explore the embedded geometrical information, rather than explicitly recovering the scene depth. We believe any image patch modeling method [63][33][63][52], will benefit from our approach. The aim of the second part of this dissertation is in the area of single image patch modeling.

In the work presented in Coates et al. [2][21], each image patch is encoded as the truncated distances with respect to the dictionary atoms constructed by simple K-means clustering or even randomly chosen training patches. Bureau et al. [17] uses random convolution filters to train a Convolution Net for object recognition. Saxe et al. [57] argues that the dictionary is not as important as had previously been thought, arguing instead that the special learning structure is the key reason for improvement. Inspired by the work of Coates et al. [2] and Zhang et al. [85], we view feature extraction from an image patch in the context of the Kernel method [61]. We argue that the feature extraction process is equivalent to the low-rank approximation of the kernel matrix using the Nystrom method. The Nystrom method is an efficient technique to generate low-rank matrix approximations from a few landmark points. By bridging the two fields and utilizing a variety of fixed and adaptive sampling schemes [74][26][56][85], explored in the Nystrom sampling field, we hope better image patch features can be learned.

Further, we propose an improved K-means clustering algorithm, GEV-Kmeans, based on the Generalized Extreme Value (GEV) distribution. Our key observation is that the squared distance of a point to its closest centroid adheres to the Generalized Extreme Value (GEV) distribution when the number of clusters is large. In contrast with the K-means algorithm, we minimize reconstruction error by ignoring those points with lower GEV probabilities (i.e. rare events), and focus on others points which might be more critical in characterizing the underlying data distribution. Consequently, our algorithm can handle outliers very well in

situations where the conventional K-means algorithm suffers.

The rest of this dissertation is organized as follows. Chapter 2 presents a novel geometry-aware image patch modeling in the context of a light-field by considering the *cross*-structure of a 2D camera array. Chapter 3 generalizes the model to leverage the full viewpoints of a 2D camera array, and applies to two applications: joint image denoising and joint angular/spatial super-resolution for a light-field. As all these applications are based on how well the EPI-patches are modeled, we explore two techniques for better image patch modeling in Chapter 4 and 5. Chapter 4 explores the connection between the K-means cluster representation [2] and Nystrem sampling techniques, and applies it to classification tasks. In Chapter 5, a new clustering method based on the Generalized Extreme Distribution is proposed. Finally, Chapter 6 summarizes this dissertation with discussions for future works.

Chapter 2

Image patch modeling in a light field

In this chapter, we consider the image patch modeling problem in the context of a light field. We exploit the scene geometric information encoded in the **patch-cube** – an image patch and its surrounding patches from multiple viewpoints – without explicitly recovering the depth of the scene. The scene geometric information embedded in Epipolar Plane Images (EPIs), including depth, depth discontinuity, occlusion, and so on, provide useful geometry aware constraints to guide our patch coding process. The applications of our EPI-based image patch model in a light field will be presented in Chapter 3.

2.1 Introduction

Dividing an image into a collection of overlapped image patches, each of which is encoded using the sparse coding model [52], achieves much success on many computer vision applications based on a single image [1], including image denoising [27][11], super-resolution [78][36], object recognition [80], and so on. However, encoding a local image patch solely based on its appearance or color, without knowing the 3D geometrical information surrounding it, renders the resulting code insufficiently stable and susceptible to shift, distortion, illumination, scale and image noise. More importantly, the scene geometry constraints might not be satisfied at all.

Different strategies are exploited to achieve relative shift or small-distortion invariance in the literature. One way is to leverage the spatial pooling, either average or max-pooling, by aggregating the codes from its spatial neighbors. Another is to enforce the codes from neighboring image patches following some human-derived priors [32][77]. Thirdly, 3D depth information, less affected by the image capture conditions, may also be considered along with image patch appearance or color [38][64][14][13][15][38]. None of these works consider the image patch representation in conjunction with scene geometric information.

Light-field cameras have emerged over the last decade as a new type of image capture device – they acquire many images simultaneously from a variety of perspectives over a

[1] We term this type of methods Image-Based Sparse Coding **IBSC** in contrast to our work.

planar imaging surface. For a given small object in 3D space, each of the cameras provides an image patch from among a 2D grid of viewpoints. These image patches are not distributed arbitrarily, they follow some constraints enforced by the design of the camera array. [8] [16] analyze the case of a 1D linear camera array, presenting correspondence-free ranging by detecting the slope present in the EPI-image, stacking all linear camera's images along the direction of camera displacement. Extending from the case of a 1D camera array, we believe the 2D camera array will provide more information about the properties of an scene. These high redundancy images provide a good opportunity to employ computer vision tasks of greater reliability and at lower cost than any collection of generally-positioned single cameras. These features of a light field camera have been exploited in image refocusing [49], depth recovery [71], and free-space computation [16]. Although much progress has been made in high quality depth recovery, it is still difficult to obtain dense depth information from a light-field camera over a large field of view.

We argue that a complete description of an image patch should include its appearance/texture, coloration, and scene geometric information. In this chapter, we will explore the image patch modeling problem by considering all this information simultaneously and in an integrated way. A geometry-aware sparse coding strategy is then presented to efficiently encode an image patch also respectful of geometric constraints embedded in a light-field, while avoiding explicit recovery of depth. To our knowledge, there are few works along this line, and none demonstrably successful.

More specifically, for an arbitrary image patch P_{rc} in the (r, c)-th camera, we begin by forming a horizontal and vertical **patch-cube**, denoted as $Q^{(h)}$ and $Q^{(v)}$, respectively. $Q^{(h)}$ is formalized by first stacking all images of the r-th row and then extracting the cube surrounding P_{rc}. $Q^{(h)}$ can be regarded in two ways: (1) as a stack of image patches, each of which is composed from different viewpoints, and (2) a stack of horizontal EPIs [16]. This duality forms the foundation of our approach – any image modeling method should simultaneously satisfy the geometric constraints implicitly embedded in the observed EPI patches. Similarly $Q^{(v)}$ can be constructed by stacking all c-th columns of the camera array around P_{rc} of interest. Further, the two **patch-cube**s are not independent to each other, but are linked by sharing the same image patch P_{rc}. Incorporating these extra implicit geometric constraints will make our image patch model more stable by explicit consideration of its spatial neighbors observed from different viewpoints. Also these geometry-related constraints will make our learned features less affected by image noise and image capturing conditions, which cause considerable degradation in the traditional image patch models operating on single images.

Our algorithm can be regarded as introducing a new set of geometric constraints into the sparse coding domain which have not been exploited before. The codes for an image patch vary and are automatically adjusted by the surrounding geometric information. These properties overcome difficulties [45] of the traditional **IBSC** model where fixed-size filters are applied over different depth levels and are incapable of modeling them all equally well. On the other hand, EPI patches are relatively simple, full of single-color continuous or broken line segments [16], and will not change dramatically for a large range of relative depth values.

They are also easier to model using the sparse coding model in contrast to the case with arbitrary image patches.

This chapter is organized as follows. In Section 2.2 we review the current related work. We present the **patch-cube** concept and its properties in Section 2.3. In Section 2.4, our geometry-aware image patch model is presented. To demonstrate the value of **patch-cube** and the light-field data, an image denoising algorithm, **NLM-LF**, is developed by simply extending the Non-Local Mean algorithm [18] using our new image patch similarity measure in Section 2.5. Experimental results and conclusions are given in Section 2.6 and 2.7, respectively.

In this Chapter, we only consider the *cross*-structure subset of a 2D camera array for a given image patch. In Chapter 3, we will take advantage of the full 2D grid structure and then apply our models to joint image denoising and angular/spatial super-resolution in a light-field camera.

2.2 Related work

In this section, we first describe some related image patch modeling methods we are going to use, and present related works within the multi-camera and light field.

Sparse-coding based image-patch modeling on a single image (**IBSC**) has been explored widely recently. Each image patch x is simple enough and can be modeled as:

$$x = D\alpha + \epsilon \tag{2.1}$$

where ϵ is additive and *i.i.d.* (independent and identically distributed) Gaussian noise, and D is an over-complete dictionary. The coefficients, α, usually very sparse, can be solved by minimizing the following objective function:

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \quad J = ||x - D\alpha|| + \lambda||\alpha||_1 \tag{2.2}$$

where λ is the Lagrange multiplier to promote the sparsity of α. The image patch can then be simply reconstructed as:

$$\hat{x} = D\alpha^* \tag{2.3}$$

In an image denoising application on a single image, the final denoised image is estimated by averaging the neighboring reconstructed image patches. The dictionary D is precomputed, which can be learned using [3][52] or any of several other methods. It has been shown experimentally that this modeling can guarantee that the dominant edges of an image are well preserved.

Another line of work is related to the similarity measure between two image patches. The representative work is non-local mean [18] for image denoising. Given a noisy image u, the goal is to recover the original image from a noisy measurement v. The pixel value $v(i)$ can be estimated as a linear combination of the neighboring pixels in the whole image:

$$\widehat{v(i)} = \sum_{j \in u} w(i,j)u(j) \tag{2.4}$$

where $w(i,j)$ measures the similarity between patch $\mathcal{N}(i)$ and $\mathcal{N}(j)$ [18]:

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{||u(\mathcal{N}_i) - u(\mathcal{N}_j)||^2}{h^2}}, \quad \text{where} \quad Z(i) = \sum_j e^{-\frac{||u(\mathcal{N}_i) - u(\mathcal{N}_j)||^2}{h^2}} \tag{2.5}$$

where $Z(i)$ is the normalizing constant and the parameter h controls the decay of the exponential function. The core of non-local mean is in measuring image patch similarity. Due to lack of scene geometrical information, the L_2 distance measurement is prone to image noise, and may also be miscalculated between *true* and *false* edges, both with similar appearance.

There have been some related works in modeling images within a light field or general camera array. For example, [40] and [47] propose that the light-field patches, from different view points, satisfy a Gaussian Mixture model conditioned on the depth, and that image denoising or spatial super-resolution be conducted by using this prior knowledge. Wanner et al. [71][72] models an image using a variational method, with the EPI-image constraints being used to drive their solution. Spatial and angular super-resolution of 4D light fields can be conducted using this variational framework. Zhang et al. [86] consider image denoising in general multi-view images by finding similar patches from multi-view images based on the depth information. However all of this work relies on the depth information of the scene being known in advance. It must be noted, though, that robust recovery of depth information over a large range of distances is not simple. In the Heidelberg work [71], for example, recovering depth from the light field uses very small baselines, necessary in order for the slopes to be well bounded, and this renders the problem very difficult when the depth range may be quite large.

In the following section, we will look at a light field, or more precisely a 2D grid camera array, and describe some of its important properties.

2.3 2D grid camera array

Our light-field data is captured using a 2D grid camera array, shown in Figure (1.1) in Chapter 1.

Given a $N \times M$ 2D camera array (4×4 in this case) $\{C_{ij}\}$(each image has resolution $S \times T$), and the reference camera denoted as $C_{\text{ref,ref}}$, any 3D point $P = (X, Y, Z)$ will form images on all cameras $\{(x_{ij}, y_{ij})\}$, where $i = 1, \ldots, N$ and $j = 1, \ldots, M$ (see Figure 2.1). The horizontal and vertical baselines for camera C_{ij} are defined as the horizontal/vertical distance between the camera centers C_{ij} and the reference camera, respectively:

$$\begin{aligned} b_{ij}^{(h)} &= ||C_{ij} - C_{i,\text{ref}}||_2 \\ b_{ij}^{(v)} &= ||C_{ij} - C_{\text{ref},j}||_2 \end{aligned} \tag{2.6}$$

From the light-field camera structure, we have the following for each camera row:

$$y_{ij} = y_{i,\text{ref}}, \quad d_{ij}^{(h)} = x_{ij} - x_{i,\text{ref}} \tag{2.7}$$

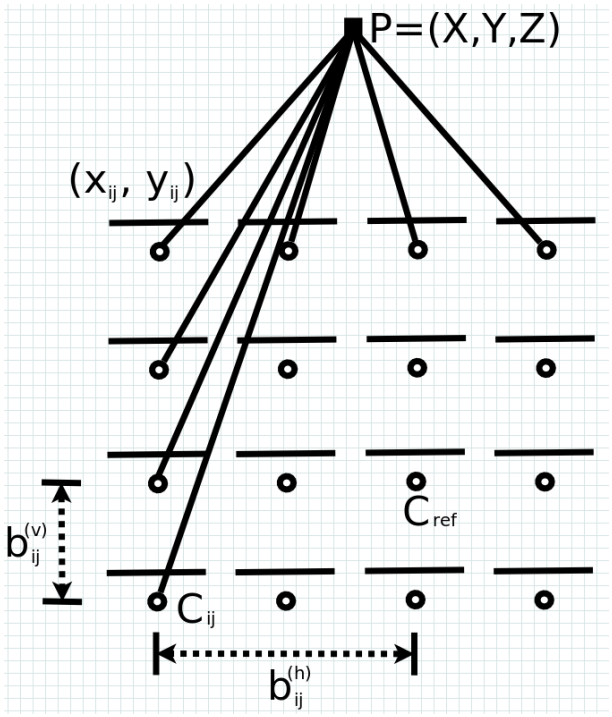


Figure 2.1: Images of a 3D point P in a 2D camera array

and similarly, for each camera column we have:

$$x_{ij} = x_{\text{ref},j}, \quad d_{ij}^{(v)} = y_{ij} - y_{\text{ref},j} \tag{2.8}$$

where $d^{(h)}$ and $d^{(v)}$ are the horizontal and vertical disparities, respectively. The Epipolar Plane property provides for the disparity varying only horizontally in each row of cameras and vertically in each column of cameras. Further, it is easy to verify the following relationship:

$$\frac{d_{ij}^{(h)}}{b_{ij}^{(h)}} = \frac{d_{ij}^{(v)}}{b_{ij}^{(v)}} \propto \frac{1}{Z} \tag{2.9}$$

This fact basically states that the horizontal and vertical disparities of P are not independent, and can be simply linked by the horizontal and vertical baseline. In comparison with the binocular stereo scenario, multiple baselines will form a richer cue for solving the correspondence problem.

A 2D grid camera system can be naively regarded as a collection of horizontal and vertical 1D linear camera arrays, which has been studied for many years. Following [8][16], we form the horizontal Epipolar Planar Image (EPI)[2], denoted as $E_{(i,y)}^{(h)}$ $(i = 1, \ldots, N)$, of size $M \times T$

[2]Here, we assume the horizontal and vertical baselines are equal, and EPIs can be formalized by stacking images together directly.

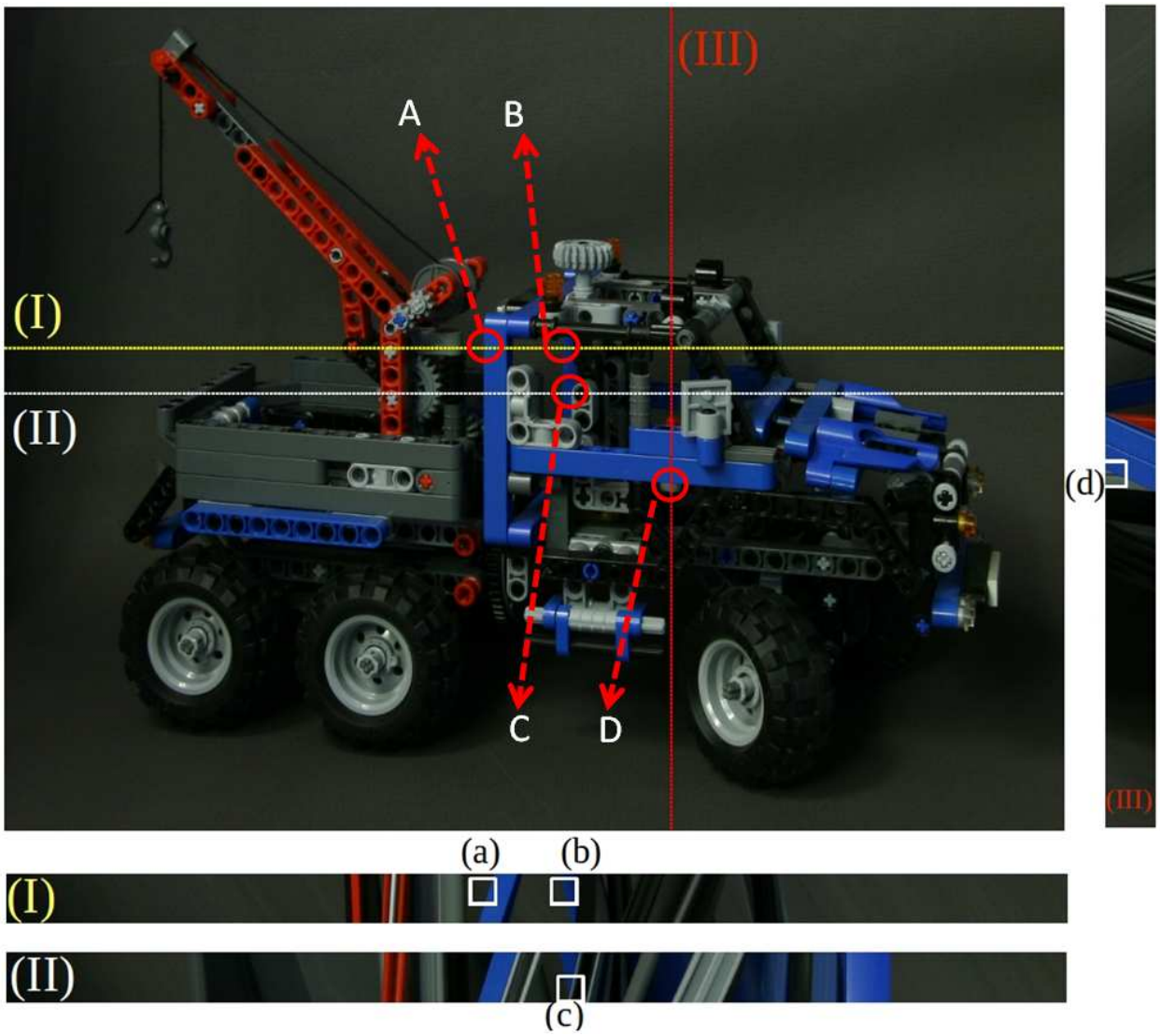


Figure 2.2: Horizontal and vertical EPI-images in a light field.

by stacking the y-th image row of all i-th camera row images (see the Fig 2.2-bottom). Similarly, the vertical EPI $E^{(v)}_{(j,x)}$ $(j = 1, \ldots, M)$ of size $S \times N$ is formed by stacking the x-th image columns of all j-th camera column images (see Fig 2.2-right). Then the first two terms in Eqn 2.9 can be measured as the slope of an EPI, which are both inversely-proportional to the depth Z [16][8].

Let's look closer at the EPI images generated horizontally and vertically, and treat an EPI as a collection of EPI-patches. Fig 2.2-bottom shows the EPIs $E^{(h)}_y$ acquired for two different rows y, labeled as (I) and (II), respectively. Fig 2.2-right shows the vertical EPI image $E^{(v)}_x$, labeled as (III). Our observations are as follows:

- EPIs are visually simpler than the original image. (1) Due to the photometric consistency condition, an EPI contains only connected or broken line segments, and the color along those segments is roughly constant. (2) The complexity of an EPI-patch will be simpler than arbitrary image patches (only rotation of lines since the depth is limited);

- The two image patches, labeled as A and B in the original image, with different depths and being located at the object boundary, can not be distinguished using their appearance only. However, their corresponding EPI patches, (a) and (b), in EPI (I) show different slopes, which clearly indicates the depth difference;

- The occlusion becomes obvious by merging different viewpoint images. The broken-line pattern in EPI-patch $(c)/(d)$ indicates that corresponding image patch $(C)/(D)$ is occluded from some viewpoints horizontally/vertically;

- For *true* edges [83], induced by the color difference on an object, each image patch (from different viewpoints) will be shifted-versions of each other, with the magnitude of the shift determined by the relative depth of this patch with respect to the viewpoint. However, the line segment pattern in both the horizontal and vertical EPI-patches should be similar, indicating the same depth;

- For *false* edges [83] induced by the depth discontinuity or occlusion, the EPI-patch contains broken-line segment patterns. The horizontal and vertical occlusion could be different. On the other hand, slight change of viewpoints will cause the image patches to differ dramatically;

- Given an image patch P from one camera, it will be part of both horizontal and vertical EPI-patches simultaneously. That is, if P is changed, it will affect both the horizontal and vertical EPI-patches.

Different from an image patch extracted from one regular image, which solely contains the appearance or color information, the EPI image provides complementary geometric information about the scene. This provides another perspective for viewing the image patch modeling problem.

Such rich geometric information in a 1D camera array has been widely used in numerous applications. Okutomi et al. [50] take advantage the adaptive baseline to balance the trade-off between precision and accuracy in stereo matching for depth recovery, using modified sum-of-squared difference (SSD) values within a lateral displacement camera array. Gelman [29] compresses multi-images using EPI. Others [16][8][71][72] recover the scene depth by detecting the slopes in EPI images while respecting the occlusion constraints.

2.4 Image patch model in a light field

Motivated by the success of the sparse coding model [52] in image patch modeling and opportunities with the 1D linear camera structure, we marry the two and aim at attaining a good image-patch model. We argue that a good image patch model should satisfy the following criteria: (1) the dominant edges in that patch be well preserved in the reconstructed version and (2) the reconstructed image patch respect the geometric constraints at that patch. The EPI patches, both horizontal and vertical, implicitly contain such geometric information and can easily fulfil the second requirement without explicit recovery of depth. In other words, after denoising in a light field, the reconstructed EPI images, both horizontal and vertical, should preserve the light field properties as much as possible (see section 2.3 for detail). Further, since the geometric information is usually less affected by image noise and lighting conditions, we expect our image patch representation to be more robust and stable.

In this section, we first present the concept of **patch-cube**, a collection of image patches acquired from different viewpoints. Then, we will show that the **IBSC** model based on a single image is not sufficient in a light field framework. Finally, we treat the EPI image as a regular image, analyze EPI patches from the sparse-coding point of view, and present our new patch-based model developed within the context of a light field.

2.4.1 The Patch Cube

Suppose we are interested in a local image patch P_0, with size of $(2w+1) \times (2h+1)$, from the (r, c)-th camera in a 2D camera array. Instead of treating this patch as stand-alone and only looking at its appearance and color, we also look at the surrounding image patches from nearby viewpoints. In this chapter, we only consider the *cross*-structure, meaning we just consider the cameras on the r-th row and the c-th column for the chosen image patch in the central view. The full 2D set of viewpoints will be modeled in the next chapter.

The horizontal **patch-cube** $Q^{(h)}$, see Figure 2.3-Left, can be constructed by stacking along the t direction all $|C|$ images in the r-th row, with size $(2w+1)\times(2h+1)\times(2t+1)$. Note that this construction is just a stacking process, we do not need the 3D depth information to localize the corresponding patches. $Q^{(h)}$ [3] can be viewed from two directions:

- Along the t-direction, $Q^{(h)}$ can be regarded as the collection of $2t+1$ **image patches**, each of which comes from one camera and has size $(2w+1) \times (2h+1)$: $Q^{(h)} = \{P^{(h)}_{-t}, \ldots, P_0, \ldots, P^{(h)}_t\}$,
- Along the y direction, we have $2h+1$ horizontal **EPI-patches** with size $(2w+1) \times (2t+1)$, [4] denoted as: $Q^{(h)} = \{E^{(h)}_{-h}, \ldots, E^{(h)}_0, \ldots, E^{(h)}_h\}$.

[3] The superscripts $\cdot^{(h)}$ and $\cdot^{(v)}$ represent the horizontal and vertical, respectively.

[4] In this presentation, we denote the **image patch** and **EPI-patch** differently.

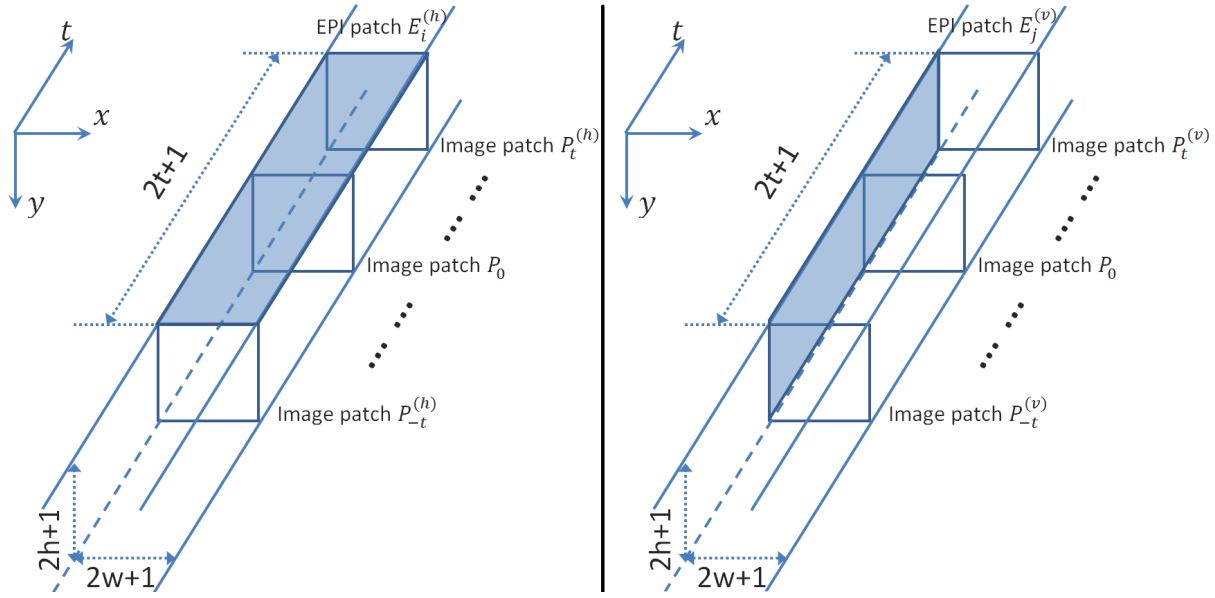


Figure 2.3: Horizontal and Vertical **patch-cube** in a light field

Similarly, the vertical **patch-cube** $Q^{(v)}$ is formed by stacking all of the corresponding patches with respect to P_0 from $2t+1$ cameras on the c-th column. $Q^{(v)}$ can be viewed either as: $\{E_{-w}^{(v)}, \ldots, E_0^{(v)}, \ldots, E_w^{(v)}\}$ or $\{P_{-t}^{(v)}, \ldots, P_0, \ldots, P_t^{(v)}\}$.

The intermingling of the EPI-patches and image patches can be further described as: [5]

$$E_i^{(h)} = \begin{pmatrix} P_{t,(i,:)}^{(h)} \\ \vdots \\ P_{-t,(i,:)}^{(h)} \end{pmatrix}, \quad E_j^{(v)} = \begin{pmatrix} P_{t,(:,j)}^{(v)} & \ldots & P_{-t,(:,j)}^{(v)} \end{pmatrix} \tag{2.10}$$

That is, the i-th horizontal EPI patch is the i-th row of all horizontal image patches, and the vertical EPI-patch is constructed as the j-th column of the vertical image patches. Also, the horizontal and vertical **patch-cube**s share the same image patch P_0 in which we are interested. That is, the central image patch P_0 can be reconstructed from both the horizontal and vertical EPI patches:

$$P_0 = \begin{pmatrix} E_{-t,(0,:)}^{(h)} \\ \vdots \\ E_{t,(0,:)}^{(h)} \end{pmatrix} = \begin{pmatrix} E_{-t,(:,0)}^{(v)} & \ldots & E_{-t,(:,0)}^{(v)} \end{pmatrix} \tag{2.11}$$

We argue that the **patch-cube** is a complete description of a local image patch: it contains enough information to describe the appearance of an image patch, and also encompasses

[5] We denote $X_{(i,:)}$ as the i-th row of a matrix X and $X_{(:,j)}$ as its j-th column.

the scene geometric information in the vicinity of that patch, embedded in the horizontal and vertical EPI-patches.

One thing worth mentioning is that the **patch-cube** is different from the regular spatial neighbors of an image patch from a single image, where each of them is solely a shifted version of the other. How to model these in a shift-invariant fashion, or how to include any priors to better capture these changes, are not clear at this point. After all, we expect our code or representation to not be affected through simply shifting by several pixels. However, putting the image patch of interest in the context of **patch-cube**, and viewing the **patch-cube** from the other direction, the resulting EPI-patches contain strong statistical regularities. These will serve as good constraints to regularize our image patch coding.

More specifically, this dual view of an image patch from its neighboring viewpoint images provides new insights on the image patch modeling problem. Due to the intermingling, modeling each image patch in a **patch-cube** is not arbitrary, but highly correlated. When we model the image patch, we want the resulting EPI-patches to have similar patterns reflecting their similar geometry. The horizontal and vertical **patch-cube**s are also not independent from each other. For example, if we presume the depth of P_0 to be roughly the same (occlusion could be slightly different from horizontal to vertical direction), we should observe similar line-patterns in EPI-patches within both **patch-cube**s.

This new type of redundant data presents a novel challenge for machine learning and computer vision researchers. In the next section, we will first show that the current image patch sparse-coding based method is not sufficient to capture this geometric information and present, in the following, the method we have developed.

2.4.2 Why modeling an image patch from a single image is not enough

In this section, we treat the **patch-cube** as a collection of regular image patches, and model them using the sparse-coding model. We inspect how the traditional **IBSC** model behaves in a light field camera by checking whether the geometrical scene information is preserved. The EPI patches, due to their properties, can serve as a good geometric measurement.

We first learn the dictionary related with the local image patches using the method described in [39], as shown in Fig 2.7-(a).[6] Then each image patch in the horizontal **patch-cube** can be reconstructed as[7]:

$$vec^{(h)}\left[\widehat{P_{t'}^{(h)}}\right] = D\alpha_{t'}^{(h)}, \quad t' = -t, \ldots, t$$

We further decompose the dictionary D into sub-blocks: $D_{(i,:)}$ is the sub-matrix relating to the i-th row of P_0. Similarly, $D_{(:,j)}$ represents the sub-matrix relating to the j-th column.

[6] To simplify the explanation, we train one single dictionary from the local patches from all images.

[7] $vec^{(h)}[\cdot]$ and $vec^{(h)}[\cdot]$ represent a matrix as a vector in row-major and color-major fashion, respectively.

The EPI-patch $E_i^{(h)}$ is composed of each i-th row of the image patch $P_{t'}^{(h)}$:

$$vec^{(h)}\left[\widehat{E_i^{(h)}}\right] = \begin{pmatrix} D_{(i,:)} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & D_{(i,:)} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & D_{(i,:)} \end{pmatrix} \begin{pmatrix} \alpha_{-t}^{(h)} \\ \alpha_{-t+1}^{(h)} \\ \vdots \\ \alpha_{t}^{(h)} \end{pmatrix} \tag{2.12}$$

Similarly, the vertical EPI-patch can be reconstructed as:

$$vec^{(v)}\left[\widehat{E_j^{(v)}}\right] = \begin{pmatrix} D_{(:,j)} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & D_{(:,j)} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & D_{(:,j)} \end{pmatrix} \begin{pmatrix} \alpha_{-t}^{(v)} \\ \alpha_{-t+1}^{(v)} \\ \vdots \\ \alpha_{t}^{(v)} \end{pmatrix} \tag{2.13}$$

Due to the properties of the **patch-cube**, we expect the reconstructed EPI-patches to satisfy: (1) sharp edges be well preserved (which indicate the scene geometric information, especially occlusions and depth discontinuities), and (2) EPI-patch patterns are preserved across image patch modeling.

From Eqn 2.12 and 2.13, we have several observations:

- For a *true* edge at non-zero disparity, the image patches are shifted versions of each other. Following [32], we assume the codes for those shifted copies of P_0 share the same sparseness pattern, which means that all $\{\alpha\}$ in Eqn 2.12 has the same non-zero elements position, then each row of the EPI-patch will be reconstructed using the same elements of the dictionary. Therefore each row of $\widehat{E_i^{(h)}}$ turns out to be roughly the same, which leads to the reconstructed EPI-patch being blurred;

- Given P_0 at a depth discontinuity, the appearance of each image patch will change dramatically due to the viewpoint change, and so do the codes generated from the sparse coding model. In general, we have nothing to predict the behavior of the reconstructed EPI-patches. Several experimental results will show that the reconstructed EPI-patches are actually blurred;

- The horizontal and vertical **patch-cube**s are not independent from each other. It is difficult for one single code for the central image patch P_0 to satisfy the EPI constraints on both **patch-cube**s.

To validate our claim experimentally, the following experiment is conducted on the **Knight** image dataset from the Stanford Light Field archive [8], see Fig-(2.4). We pick a 7×7 patch, P_0, in the central view, $(9, 9)$-th camera, from the 2D camera array, shown in Fig-(2.4)-(a). Notice this image patch locates on the object boundary, half of which is about

[8] See Section 2.6 for details

the background. Our goal is to check whether the traditional **IBSC** can still preserve the geometry information embedded in the EPI-patches. The top-row of Fig-(2.4)-(b), (c), (d) and (e) show the horizontal and vertical image patches around P_0, and the horizontal and vertical EPI-patches, respectively. The middle-row shows the results from the **IBSC** model: we model each image patch separately using sparse coding model, the reconstructed image patches are shown in the middle-row of Fig-(2.4)-(b), (c). Also the resulting reconstructed EPI-patches are shown in the middle-row of Fig-(2.4)-(d) and (e). From the figure, we can see that the **IBSC** model can blindly and faithfully reconstruct the input image patches, however serious blur occur in the resulting EPI-patches. Since the patterns in EPI-patches are related to scene geometry, this suggests the **IBSC** model cannot respect image geometric structure. More experimental results will be given in the Experimental section.

The fact that the **IBSC** model only considers the appearance and color will worsen the problem. Given several image patches with similar appearance/color, but differing depth, **IBSC** model will encode these patches similarly. Due to the intermingling between image patches and EPI-patches, one single set of codes is hard to satisfy all geometric constraints at different geometric surroundings, depth, for example. Also, the **IBSC** model is more affected by image noise, lighting conditions, etc.

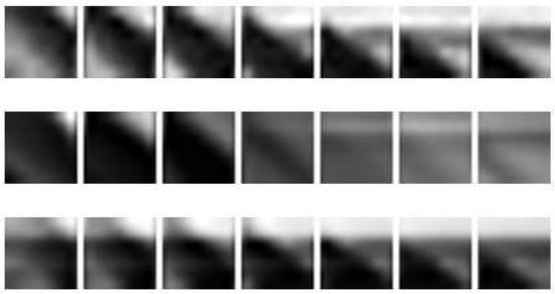
2.4.3 Motivation

In comparison with directly modeling the image patches, modeling the EPI-patches provides the following benefits: First, the EPI-patch contains scene geometric information, including depth of the patch and any occlusion information, etc. Therefore, it is less affected by the image capture conditions. Second, the EPI-patch consists of only lines and line segments, and the color along such line segments is mostly uniform, which make this EPI-patch simpler than an arbitrary natural image patch. More importantly, no matter whether P_0 is a true edge or located at a depth discontinuity, the EPI patches will be similar from the perspective of geometry, where it is easier to pose priors or constraints for our problem. In contrast, the **IBSC** model has difficulties to compensate dramatic image patch appearance change due to viewpoint change, especially at occlusion or depth discontinuities. Knowing these will also help in object segmentation and performing spatial pooling in object recognition.

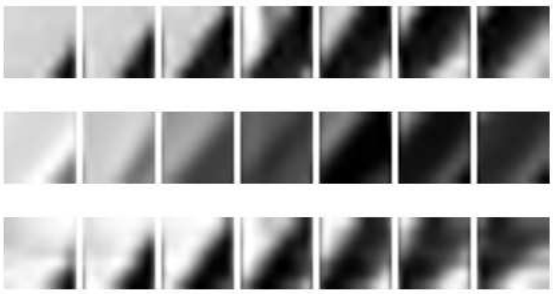
To validate our ideas and compare with the **IBSC** model, we continue the experiments conducted in Section 2.4.2 (see Fig (2.4)).[9] Instead of modeling the image patches in a **patch-cube** as the **IBSC** model, we directly apply the sparse coding model on the horizontal and vertical EPI-patches (top row in Fig-(2.4)-(d) and (e)), respectively. The reconstructed EPI-patches are shown in the bottom row of Fig-(2.4)-(d) and (e), respectively, and further the resulting reconstructed image patches are shown in the bottom row of Fig-(2.4)-(b) and (c). To make the comparison more clear, Fig-(2.4)-(f) shows in order: the original image patch from the $(9, 9)$-th camera, the horizontally reconstructed, vertically reconstructed image

[9]The more complete algorithm follows later.

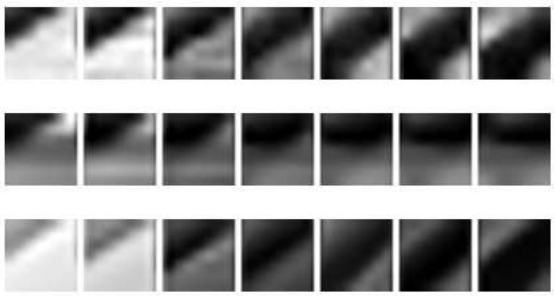
(a) Target image patch(blue square)



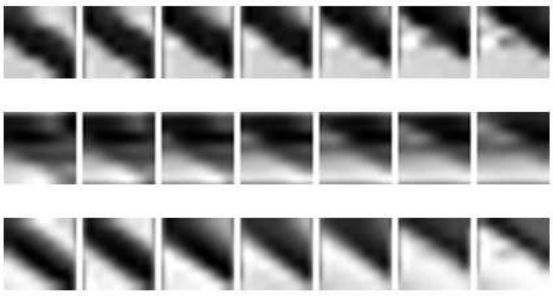
(b) Horizontal image patches



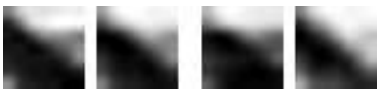
(c) Vertical image patches



(d) Horizontal EPI patches



(e) Vertical EPI patches



(f) Our final reconstructed image patches

Figure 2.4: Reconstruction of image patch VS EPI-patch. Original image patch shown in (a). See Sec.2.4.2 and 2.4.3 for details.

patches, and our final reconstructed image patch of P_0. The final one is just an average of the two.

From the figure, we can see that: 1) the sharp edges in both horizontal and vertical EPI-patches are all preserved, and 2) we can also faithfully reconstruct the original image patch P_0. To make the comparison fair, both **IBSC** model and our model use roughly the same number of non-zero elements for the sparse coefficients.

This scene geometrical information, such as depth and occlusion, are automatically embedded in EPI images. By analysing this information, without explicitly recovering the depth, we will gain structural insights without the cost and risk of range computation. Our goal is different from that of **IBSC**: faithfully reconstruct the image patch and preserve the sharp edges that are present. A further goal of ours is to ensure the reconstructed image patch respects the scene structural information, including depth and occlusions, while ensuring that the horizontal/vertical structural information is consistent.

Observing the power of the **patch-cube**, we will explore it in the context of joint image denoising in a light field. In this following section, 2.4.4, our new image patch model in a light field camera will be presented. In Section 2.5, we also show that the **patch-cube** idea can boost performance of the Non-Local Mean denoising method. In this chapter, we consider only the *cross*-structure in a light field. In Chapter 3, we will further extend our **patch-cube** concept to the 2D grid, instead of the *cross* structure in a 2D camera array.

2.4.4 Scene geometry-aware image patch modeling

We seek a simple formulation to explore the solution space: given a light field and an image patch P_0, and the associated horizontal/vertical **patch-cube**: $Q^{(h)}$ and $Q^{(v)}$, our goal is simply to model this image patch so that geometrical information embedded in the light field is respected.

To simplify notation, we assume the **patch-cube** has equal dimension $n \times n \times n$, and the dictionary has size $d \times N$, where $d = n^2$ and $N >> d$. The horizontal and vertical dictionary, $D^{(h)}$ and $D^{(v)}$, have been trained (see Section 2.4.7 for further details.)

Generative model for image patch in a light-field

Given an image patch P_0, we are not interested in directly modeling its appearance/intensity alone. Instead we will look at the **patch-cube**s surrounding it, which can located at an arbitrary image in a 2D camera array.

We treat each EPI-patch in a horizontal **patch-cube** as a regular image patch and model it as:

$$vec^{(h)}(E_i^{(h)}) = D^{(h)}\alpha_i + \epsilon, \quad i = 1, \ldots, n \tag{2.14}$$

Similarly, each EPI-patch in the vertical **patch-cube** can be modeled as:

$$vec^{(v)}(E_j^{(v)}) = D^{(v)}\beta_j + \epsilon, \quad j = 1, \ldots, n \tag{2.15}$$

where ϵ is additive and $i.i.d$ white noise. $D^{(h)} \in \mathbb{R}^{d\times N}$ and $D^{(v)} \in \mathbb{R}^{d\times N}$ are the dictionaries precomputed on the horizontal and vertical EPI-patches, respectively.[10]. Similar to the **IBSC** model, we assume $d << N$ and both of the codes, $\{\alpha_i\}$ and $\{\beta_j\}$, contain many zeros. By enforcing coefficients α_i and β_j to be sparse, we can preserve sharp edges in the EPI-patches, both horizontally and vertically. This is precisely what we want in preserving scene geometric information. Further, since the EPI-patch is usually more simple than the regular image patch, we hope greater sparsity can be achieved.

After finding the optimal sparse coding of $\{\alpha_i\}$ and $\{\beta_j\}$, the target patch P_0 can then be reconstructed from horizontal and vertical directions, respectively. We re-organize the horizontal and vertical dictionaries as stacks of the following sub-matrices:

$$D^{(h)} = \begin{bmatrix} D^{(h)}_{(1,:)} \\ D^{(h)}_{(2,:)} \\ \vdots \\ D^{(h)}_{(n,:)} \end{bmatrix}_{d\times N}, \quad D^{(v)} = \begin{bmatrix} D^{(v)}_{(:,1)} \\ D^{(v)}_{(:,2)} \\ \vdots \\ D^{(v)}_{(:,n)} \end{bmatrix}_{d\times N} \tag{2.16}$$

then the patch P_0 can be reconstructed as: [11]

$$\widehat{P^{(h)}_0} = \begin{bmatrix} \left(D^{(h)}_0\alpha_1\right)^T \\ \left(D^{(h)}_0\alpha_2\right)^T \\ \vdots \\ \left(D^{(h)}_0\alpha_n\right)^T \end{bmatrix} = \begin{bmatrix} (\alpha_1)^T \\ (\alpha_2)^T \\ \vdots \\ (\alpha_n)^T \end{bmatrix} \left(D^{(h)}_0\right)^T \tag{2.17}$$

and

$$\widehat{P^{(v)}_0} = \begin{bmatrix} D^{(v)}_0\beta_1 & D^{(v)}_0\beta_2 & \dots & D^{(v)}_0\beta_n \end{bmatrix} = D^{(v)}_0 \begin{bmatrix} \beta_1 & \beta_2 & \dots & \beta_n \end{bmatrix} \tag{2.18}$$

That is, the image patch P_0 can be reconstructed by collecting either all 0-th rows of the reconstructed horizontal EPI-patches of $\{\widehat{E^{(h)}_i}\}$, or the 0-th columns of the vertical EPI-patches $\{\widehat{E^{(v)}_j}\}$.

Therefore, our final generative process for the target image patch P_0 can be represented as the following by combining Eqn 2.17 and Eqn 2.18:

$$P_0 = \frac{1}{2}\left(\widehat{P^{(h)}_0} + \widehat{P^{(v)}_0}\right) + \epsilon \quad = \begin{bmatrix} (\alpha_1)^T \\ (\alpha_2)^T \\ \vdots \\ (\alpha_n)^T \end{bmatrix} \left(D^{(h)}_0\right)^T + D^{(v)}_0 \begin{bmatrix} \beta_1 & \beta_2 & \dots & \beta_n \end{bmatrix} + \epsilon \tag{2.19}$$

[10] The property will be given later.

[11] To simplify notation, we assume the sub-matrices related to P_0 are labelled $D^{(h)}_0$ and $D^{(v)}_0$.

Based on Eqn 2.19, we assume our patch P_0 is generated from horizontal and vertical directions separately, and also the coefficients $\{\alpha_i\}$ and $\{\beta_j\}$ satisfy the following distribution:

$$\alpha_i \sim e^{-|\alpha_i|_1}, \quad \beta_j \sim e^{-|\beta_j|_1}$$

and then average. The average will ensure the final cleaned image will be consistent with both the horizontal and vertical reconstructions. This constraint provides excellent priors on how to choose the optimal representation in a sparse coding model.

Therefore, the objective function can be written as:

$$\{\{\alpha_i^*\},\{\beta_j^*\}\} = \underset{\{\alpha_i\},\{\beta_j\}}{\operatorname{argmin}} \quad \left|\left|P_0 - \frac{1}{2}\left(\widehat{P_0^{(h)}} + \widehat{P_0^{(v)}}\right)\right|\right|_{Fro}^2 \tag{2.20}$$

$$s.t.: \quad \{\alpha_i^*\} = \underset{\{\alpha_i\}}{\operatorname{argmin}} \quad \sum_i \left[\left|\left|E_i^{(h)} - D^{(h)}\alpha_i\right|\right|_2^2 + \lambda\left|\left|\alpha_i\right|\right|_1\right] \tag{2.21}$$

$$\{\beta_j^*\} = \underset{\{\beta_j\}}{\operatorname{argmin}} \quad \sum_j \left[\left|\left|E_j^{(v)} - D^{(v)}\beta_j\right|\right|_2^2 + \eta\left|\left|\beta_j\right|\right|_1\right] \tag{2.22}$$

$||\cdot||_{Fro}$ is the Frobenius norm of a matrix. Where $\widehat{P_0^{(h)}}$ and $\widehat{P_0^{(v)}}$ are the reconstructed versions of the original patch P_0 from horizontal and vertical **patch-cube**s, respectively. The first constraint in Eqn 2.21 relates to sparse coding of horizontal EPI-patches, and the second constraint in Eqn 2.22 relates to sparse coding of vertical EPI-patches. These two terms aim at getting the model on all EPI-patches to preserve sharp edges there, which is necessary in preserving scene geometric information. These two processes are linked through the reconstruction error of the final image patch $\widehat{P_0}$ (see Eqn 2.20), in order to guarantee the two reconstructions are in agreement.

The cost function in Eqn 2.20 serves as a consistency measure between horizontal and vertical reconstructions. Different objective can be used here, for example we can use l_1-type of cost measure to make the optimization solution more robust to noise:

$$\left|\left|P_0 - \frac{1}{2}\left(\widehat{P_0^{(h)}} + \widehat{P_0^{(v)}}\right)\right|\right|_{l_1}$$

In the following section, we present our iterative algorithm to solve this optimization problem, followed by our experimental results.

2.4.5 Optimization

The optimization problem in Eqn. 2.20 is equivalent to the following:

$$\begin{aligned}\min_{\{\alpha_i\},\{\beta_j\}} & \left|\left|P_0 - \frac{1}{2}\left(\widehat{P_0^{(h)}} + \widehat{P_0^{(v)}}\right)\right|\right|_F^2 \\ & + \gamma \sum_i \left[\left|\left|E_i^{(h)} - D^{(h)}\alpha_i\right|\right|_2^2 + \lambda \left|\left|\alpha_i\right|\right|_1\right] \\ & + \zeta \sum_j \left[\left|\left|E_j^{(v)} - D^{(v)}\beta_j\right|\right|_2^2 + \eta \left|\left|\beta_j\right|\right|_1\right] \end{aligned} \tag{2.23}$$

where γ and ζ are Lagrange multipliers to control the contributions of Eqn 2.21 and 2.22, respectively.

Optimization of Eqn 2.23 is difficult due to the coupling between $\{\alpha\}$ and $\{\beta\}$. Therefore, in an alternating fashion, we can solve two convex optimizations both as l_1-regularized least squares problems: solving for the coefficients $\{\alpha_i\}$ keeping $\{\beta_j\}$ fixed, and solving for $\{\beta_j\}$ keeping $\{\alpha_i\}$ fixed.

Horizontal Sparse Coding

Suppose we fix $\{\beta_j\}$, α_i can be solved by minimizing:

$$\alpha_i^* = \underset{\alpha_i}{\operatorname{argmin}} \quad \gamma \underbrace{\left[\left|\left|vec\left[E_i^{(h)}\right] - D^{(h)}\alpha_i\right|\right|_2^2 + \lambda|\alpha_i|_1\right]}_{①} + \underbrace{\left|\left|\left(A_0^{(i,:)}\right)^T - \frac{1}{2}D_0^{(h)}\alpha_i\right|\right|_2^2}_{②} \tag{2.24}$$

where the **horizontal residual image** A_0 is defined as:

$$A_0 \coloneqq P_0 - \frac{1}{2}\widehat{P_0^{(v)}} \tag{2.25}$$

and $A_0^{(i,:)}$ is the i-th row of A_0. The part labeled ① is the same as the sparse coding modeling of the EPI-patch $E_i^{(h)}$, which determines sparse coefficients for reconstructing $E_i^{(h)}$. However, the ② part will further force its 0-th row (corresponding to the i-th row of P_0) reconstruction to be close to the *horizontal residual image* A_0. Essentially, we want to put more weight on the 0-th row of $E_i^{(h)}$ and re-distribute the non-zero elements of α_i such that not only the EPI-patch $E_i^{(h)}$ can be well reconstructed but also the i-th row of the resulting image patch $\widehat{P_0^{(h)}}$ to be consistent with the vertical reconstruction.

In order to solve the optimization in Eqn 2.24, we propose two different approaches. The first approach proceeds by augmenting the original EPI-patch vector to include the residual

vector $A_0^{(i,:)}$. Eqn 2.24 can be re-written as:

$$\alpha_i^* = \underset{\alpha_i}{\operatorname{argmin}} \quad J = \gamma \left\| \begin{bmatrix} vec\left[E_i^{(h)}\right] \\ \left(A_0^{(i,:)}\right)^T \end{bmatrix}_{(n+d)\times 1} - \begin{bmatrix} D^{(h)} \\ \frac{1}{2}D_0^{(h)} \end{bmatrix}_{(n+d)\times N} \alpha_i \right\|_2^2 + \lambda|\alpha_i|_1 \tag{2.26}$$

Eqn 2.26 can be solved by various methods, Orthogonal Matching Pursuit (OMP), [39], etc. Another approach is to take gradients with respect to α_i as in Eqn 2.27 and apply a gradient descent optimization:

$$\begin{aligned} \frac{\partial J}{\partial \alpha_i} &= 2\left[\left(D_0^{(h)}\right)^T D_0^{(h)} + \gamma\left(D^{(h)}\right)^T D^{(h)}\right]\alpha_i \\ &\quad - \left[2\gamma\left(D^{(h)}\right)^T E_i^{(h)} + \left(D_0^{(h)}\right)^T \left(A_0^{(i,:)}\right)^T\right] + \gamma\lambda\frac{\partial|\alpha_i|}{\partial\alpha_i} \end{aligned} \tag{2.27}$$

where $\frac{\partial|\alpha_i|}{\partial\alpha_i}$ is an all-zero vector except the position corresponding to the largest absolute value, which equals to the sign of that element.

Algorithm 1 Horizontal sparse coding

Input: $D^{(h)}$ ($D_0^{(h)}$), $E_i^{(h)}$ and $A_0^{(i,:)}$
Output: α_i

1: Argument $D^{(h)}$ using $D_0^{(h)}$, $E_i^{(h)}$ uing $A_0^{(i,:)}$ as in Eqn 2.26
2: Solve Eqn 2.26 using OMP or [39]

Our algorithm is summarized in Algorithm 1.

Vertical Sparse Coding

In a similar manner, we perform the **vertical sparse coding** by fixing $\{\alpha_i\}$. The *vertical residual image* B_0 is defined as:

$$B_0 \coloneqq P_0 - \frac{1}{2}\widehat{P_0^{(h)}} \tag{2.28}$$

and the j-th column of B_0 as $B_0^{(:,j)}$, then we can solve β_j by minimizing:

$$\beta_j^* = \underset{\beta_j}{\operatorname{argmin}} \quad \zeta \underbrace{\left[\left|\left|E_j^{(v)} - D^{(v)}\beta_j\right|\right|_2^2 + \eta|\beta_j|_1\right]}_{①} + \underbrace{\left|\left|B_0^{(:,j)} - \frac{1}{2}D_0^{(v)}\beta_j\right|\right|_2^2}_{②} \tag{2.29}$$

Similarly, we want to focus more on the 0-th column of the EPI-patch $E_j^{(v)}$, in order to make the horizontal and vertical reconstructions of P_0 consistent.

As in the previous section, the β_j can be solved by the augmented sparse coding model Eqn 2.30:

$$\beta_j^* = \underset{\beta_j}{\operatorname{argmin}} \quad J = \zeta \left\| \begin{bmatrix} E_j^{(v)} \\ B_0^{(:,j)} \end{bmatrix}_{(n+d)\times 1} - \begin{bmatrix} D^{(v)} \\ \frac{1}{2} D_0^{(v)} \end{bmatrix}_{(n+d)\times N} \beta_j \right\|_2^2 + \eta |\beta_j|_1 \tag{2.30}$$

or gradient descent algorithm using the gradient of J with respect to β_j:

$$\begin{aligned} \frac{\partial J}{\partial \beta_j} = & 2 \left[\left(D_0^{(v)} \right)^T D_0^{(h)} + \zeta \left(D^{(v)} \right)^T D^{(v)} \right] \beta_j \\ & - \left[2\zeta \left(D^{(v)} \right)^T E_j^{(v)} + \left(D_0^{(v)} \right)^T B_0^{(:,j)} \right] + \zeta \eta \frac{\partial |\beta_j|}{\partial \beta_j} \end{aligned} \tag{2.31}$$

The algorithm is summarized in Algorithm 2:

Algorithm 2 Vertical sparse coding

Input: $D^{(v)}$ ($D_0^{(v)}$), $E_j^{(v)}$ and $B_0^{(:,j)}$
Output: β_j

1: Argument $D^{(v)}$ using $D_0^{(v)}$, $E_j^{(v)}$ uing $B_0^{(:,j)}$ as in Eqn.2.30
2: Solve Eqn.2.30 using OMP or [39]

Final algorithm for image patch modeling

Our final iterative optimization procedure proceeds as in Algorithm 3. Generally speaking, our algorithm iterates between horizontal and vertical reconstructions of the patch in which we are interested, not only preserving the sharp edges in all EPI-patches in the **patch-cube**, but also ensuring that the horizontal/vertical reconstructions are in agreement with each other. In practice, our algorithm stabilizes within a few iterations.

Our setting is not particularly sensitive to the initialization of $\{\alpha_i\}$ or $\{\beta_j\}$. Taking α_i as an example, we can either solve for Eqn 2.24-① using l_2 regression by ignoring the l_1 constraint, or directly employ sparse coding to get the $\{\alpha_i\}$.

2.4.6 Another perspective

Let's revisit Eqn 2.24 to get deeper insight into our algorithm. First, we will re-write it as: (to simplify the notations, we drop γ)

Algorithm 3 Modeling image patch P_0 in a light-field

Input: $D^{(h)}$, $D^{(v)}$, $\{E_j^{(v)}\}$, $\{E_i^{(h)}\}$ and P_0

Output: $\widehat{P_0}$

```
Initialize {β_j};
while not converge do
    Calcuate vertical residual image B_0 using Eqn 2.28;
    for each row i = 1 : n do
        Update α_i using Algorithm 1
    end for
    Calculate horizontal residual image A_0 using Eqn 2.25;
    for each column j = 1 : n do
        Update β_j using Algorithm 2
    end for
end while
Calculate P̂_0 using Eqn 2.19.
```

$$
\begin{aligned}
J_1 &= \left[\left|\left| vec^{(h)}\left[E_i^{(h)}\right] - D^{(h)}\alpha_i \right|\right|_2^2 + \left|\left| \left(A_0^{(i,:)}\right)^T - \frac{1}{2} D_0^{(h)}\alpha_i \right|\right|_2^2 \right] + \lambda|\alpha_i|_1 \\
&= \left|\left| diag\begin{pmatrix} 1 & 1 & \dots & 1+w_i^{(h)} & \dots & 1 \end{pmatrix} \times \begin{pmatrix} E_{i,(1,:)}^{(h)} - D_{(1,:)}^{(h)}\alpha_i \\ E_{i,(2,:)}^{(h)} - D_{(2,:)}^{(h)}\alpha_i \\ \vdots \\ E_{i,(0,:)}^{(h)} - D_0^{(h)}\alpha_i \\ \vdots \\ E_{i,(n,:)}^{(h)} - D_{(n,:)}^{(h)}\alpha_i \end{pmatrix} \right|\right|_F^2 + \lambda|\alpha_i|_1
\end{aligned} \tag{2.32}
$$

where the weight $w_i^{(h)}$ is defined as the following:

$$
w_i^{(h)} = \frac{\left|\left| \left(A_0^{(i,:)}\right)^T - \frac{1}{2} D_0^{(h)}\alpha_i \right|\right|_2^2}{\left|\left| E_{i,(0,:)}^{(h)} - D_0^{(h)}\alpha_i \right|\right|_2^2} \tag{2.33}
$$

where $E_{i,(k,:)}^{(h)} \in \mathbb{R}^{n\times 1}$ represents the k-th row of the EPI-patch $E_i^{(h)}$. In contrast with treating each EPI-patch as a regular image and then applying the sparse coding model, we put more weight on the row of $E_i^{(h)}$ which relates to the image patch P_0 where we are interested. The amount of weight depends on how well the vertical **patch-cube** can approximate the image patch P_0. Therefore, each iteration process, both horizontal and vertical sparse coding

steps, can be treated as a weighted sparse coding strategy. The weights are automatically determined and adjusted by the reconstruction error associated with the reconstruction of the other direction.

Similarly, for the vertical reconstruction stage, the cost function in Eqn 2.29 can be rewritten as:

$$
\begin{aligned}
J_2 &= \left[\left|\left| vec^{(c)}(E_j^{(v)}) - D^{(v)}\beta_j \right|\right|_2^2 + \left|\left| B_0^{(:,j)} - \frac{1}{2} D_0^{(v)}\beta_i \right|\right|_2^2 \right] + \lambda|\beta_i|_1 \\
&= \left|\left| diag\begin{pmatrix} 1 & 1 & \dots & 1+w_j^{(v)} & \dots & 1 \end{pmatrix} \times \begin{pmatrix} E_{j,(:,1)}^{(v)} - D_{(:,1)}^{(v)}\beta_i \\ E_{i,(:,2)}^{(v)} - D_{(:,2)}^{(v)}\beta_i \\ \vdots \\ E_{i,(:,0)}^{(v)} - D_0^{(v)}\beta_i \\ \vdots \\ E_{i,(:,n)}^{(v)} - D_{(:,n)}^{(v)}\beta_i \end{pmatrix} \right|\right|_F^2 + \lambda|\beta_j|_1
\end{aligned} \tag{2.34}
$$

where the weight $w_j^{(v)}$ is defined as the following:

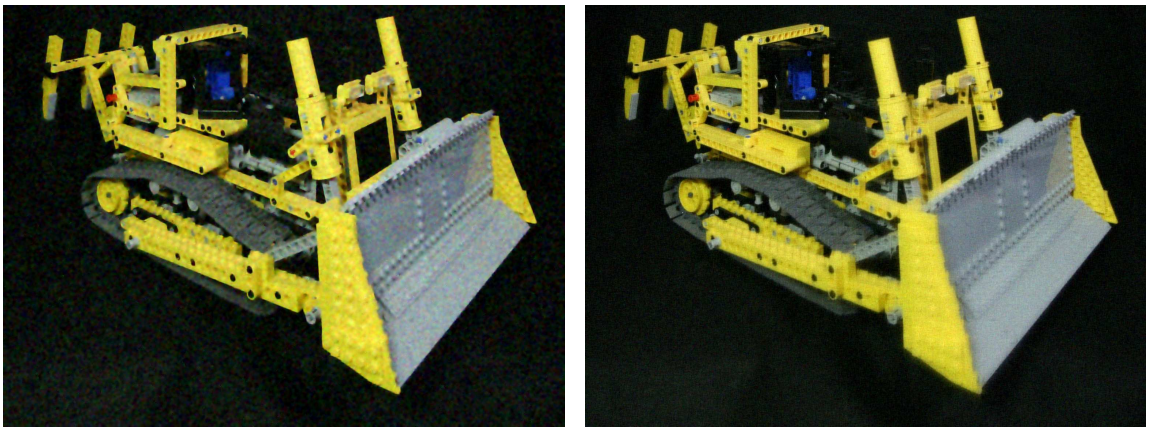
$$
w_j^{(v)} = \frac{\left|\left| B_0^{(:,i)} - \frac{1}{2} D_0^{(v)}\beta_i \right|\right|_2^2}{\left|\left| E_{j,(:,0)}^{(v)} - D_0^{(v)}\beta_i \right|\right|_2^2} \tag{2.35}
$$

From Eqn 2.32 and 2.34, we can see that: if the horizontal and vertical weights w are equal to zeros, then our algorithm will be equivalent to separately applying the sparse coding model on horizontal and vertical EPI-patches, without knowing the other. This is actually the simplified algorithm we used in Section 2.4.3. On texture-less regions, the weights, both horizontal and vertical, will be small. However, if strong texture changes or on object boundaries, our algorithm will then dynamically adjust the weights such that those regions can be well described to comply to the scene geometry due to multiple viewpoints. Usually, the horizontal weights will be bigger at the horizontal depth discontinuities and the vertical weights larger at the vertical depth discontinuities.

In order to further analyze algorithm performance, we apply it on the Stanford Light Field dataset, using the **Lego-Bulldozer** image dataset [12]. All of the light-field images are polluted by white noise with standard deviation $\sigma = 75$ pixels as our inputs. We evaluate how the following related quantities evolves before and after optimization, see Fig 2.5:

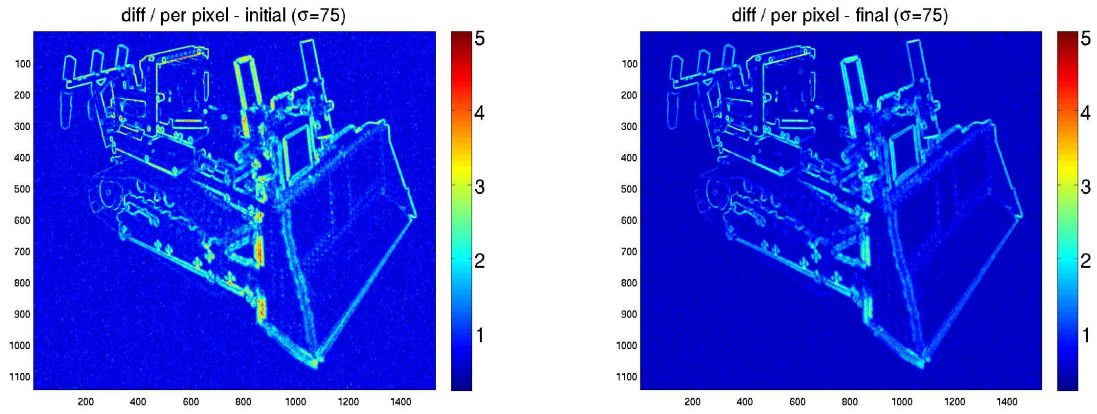
- The reconstruction error $|I - \hat{I}|$, where I and $\hat{I}$ are the original image and the denoised image. Roughly this reflects our consistency measure $P_0 - \frac{1}{2}\left(\widehat{P_0^{(h)}} - \widehat{P_0^{(v)}}\right)$ in Eqn 2.20;

[12] See Section 2.6 for details



(a) ksvd denoised result

(b) our result



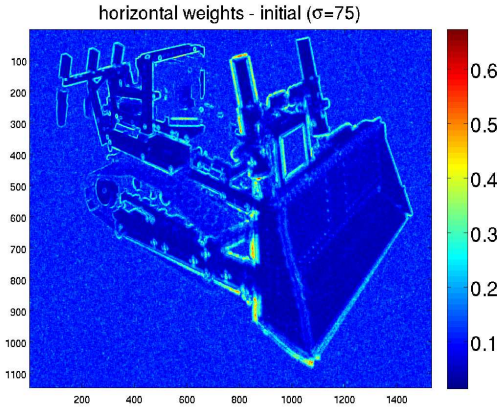
(c) recon. error - init

(d) recon. error - optim.

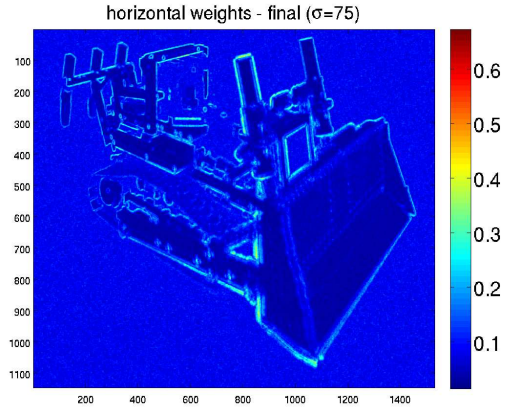
Figure 2.5: Comparison of reconstruction error, horizontal and vertical weights: before and after optimization. See the text for details.

- The horizontal weights and vertical weights: $w_i^{(h)}$ (Eqn 2.33) and $w_j^{(v)}$ (Eqn 2.35).

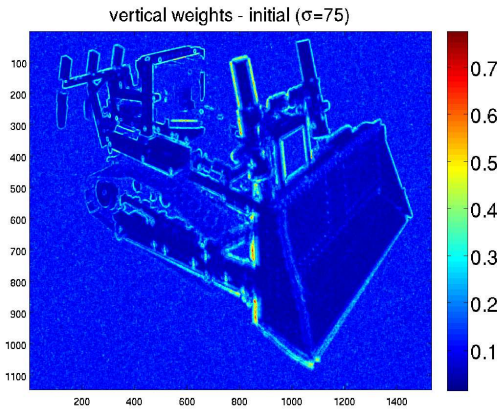
The Fig (2.5)-(a) and (b) show the denoised result comparison between [27] and our algorithm, we can see that our algorithm performs much better in locations of depth discontinuity. The rest of the sub-figures show how our algorithm performs before and after optimization. Larger pixel discrepancies are observed at depth discontinuities while relatively small discrepancy is observed elsewhere. In comparison with Fig 2.5-(c), Fig 2.5-(d) significantly reduces the errors at those discontinuities and detail there seems reasonably well preserved. In the initialization stage, see Fig (2.5)-(e) and (g), we observe large horizontal and vertical weights value on horizontal and vertical discontinuity, respectively. In the homogeneous regions without too much texture, smaller weights are observed. Our algorithm will more focus on those regions with larger weights, where scene geometric information from



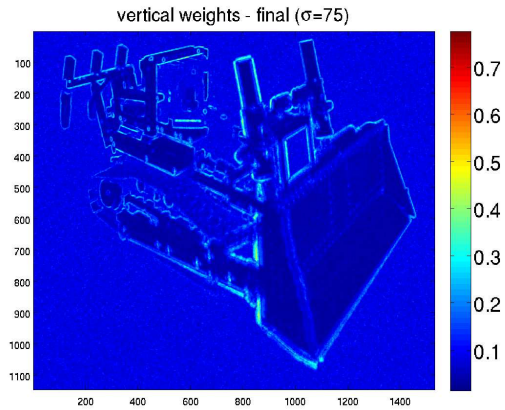
(e) hori. weights - init



(f) hori. weights - optim.



(g) vert. weights - init



(h) vert. weights - optim.

Figure 2.5: (continued)Comparison of reconstruction error, horizontal and vertical weights: before and after optimization. See the text for details.

multiple viewpoints will play more important roles. The resulting weights, both horizontal and vertical, will be efficiently reduced, see Fig (2.5)-(f) and (h), which leads to the reconstruction error at those regions dropping, see Fig 2.5-(d). One surprising observation is that our algorithm also performs better in texture-less regions. The reason is simple: although the EPI-patches contain little obvious pattern there, the simple averaging operation across multiple viewpoints can effectively reduce the noise. On the other hand, the noise will be coded equally in the traditional **IBSC** model, and then the resulting codes will be more affected by texture-less noise.

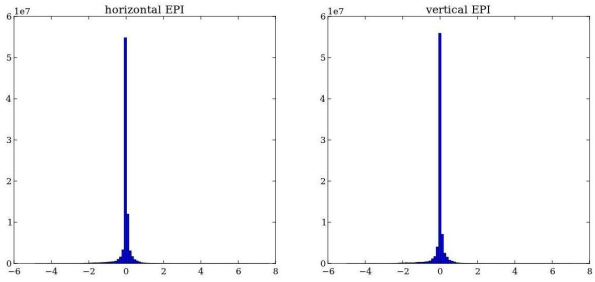


Figure 2.6: DCT responses of hori./vert. EP-I-edges on DCT basis

2.4.7 Discussion

In this section, we will discuss some related problems for our algorithm.

Dictionary

The above discussion assumed the dictionaries, $D^{(h)}$ and $D^{(v)}$, had been precomputed. In this section, we briefly discuss our process.

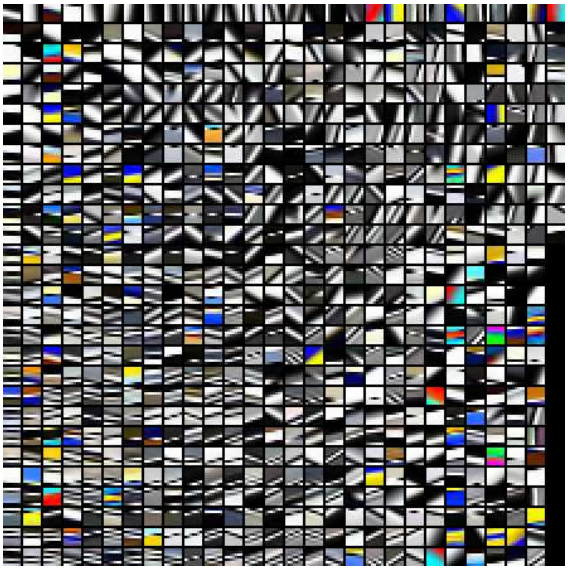
First, we collect numerous horizontal and vertical EPI-patches from the Stanford light-field dataset. Fig 2.6 shows the response histogram of these EPI-patches with respect to the standard DCT basis. The heavy tails are observed for both horizontal and vertical EPI-patches, and there seems not to be much difference between the two.

Our dictionaries are trained for horizontal and vertical directions separately, using the method presented in [27] [13]. Also we do not train on R,G and B channels separately, instead we train a single dictionary by concatenating R, G and B channels as a single vector, in order to avoid the false color phenomena. We visualize the learned dictionaries from EPI-patches, both horizontal and vertical (see Fig 2.7b and 2.7c, respectively). The dictionary size is chosen as 1000. As a comparison, we also show the dictionary atoms learned from the image patches, see Fig 2.7a.

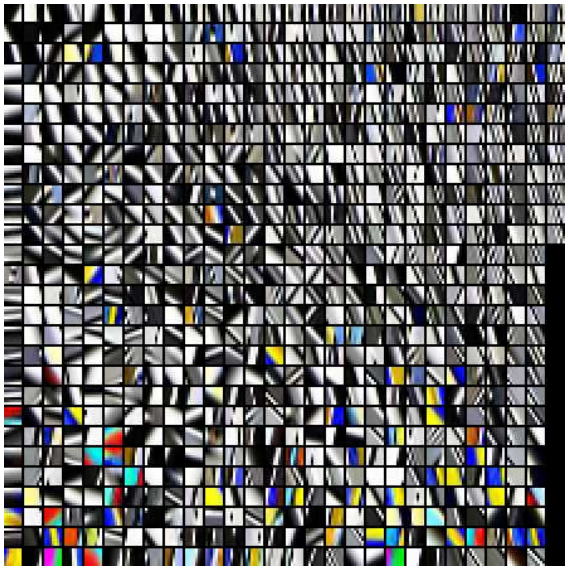
From the above figures, we can see that all of the dictionary atoms resemble sharp edges in the patches. However the dictionary atoms in our EPI-patches are more meaningful: these edges are actually the slope of the EPI-images, representing the actual depth of that atom. Further, the horizontal and vertical dictionaries seem different in capturing the horizontal and vertical edges, which was our expectation. We believe the edge patterns, including occlusion, are different in the natural world. For example, vertical edges will be more numerous than horizontal edges. Further, due to the properties of EPI-images, having near single color, and only line segments or broken segments, the EPI-patch patterns are much simpler in our EPI-dictionaries than those in a regular dictionary. This suggests smaller dictionary sizes

[13] Other methods can also applied here. In later chapters, our method based on KPCA and GEV is presented.

(a) Dictionary on image patches



(b) Dictionary on horizontal EPI patches



(c) Dictionary on vertical EPI patches

Figure 2.7: Comparison of leaned dictionaries from image patches and horizontal/vertical EPI images.

will be sufficient for our modeling problems. Investigating the impact of dictionary size on the modeling performance will be left as future work.

Data normalization

Similar to [27], we remove the DC components before applying our model. In distinction, we keep track of the DC component separately for both horizontal and vertical directions. During our modeling stage, the reconstruction error is actually measured by putting the horizontal and vertical DC components back into the horizontal and vertical reconstructions, respectively. This makes the consistency comparison between horizontal and vertical reconstructions easier, see Eqn 2.19.

Edge preservation and expressive power

Given an image patch P_0, the sharp edges on both directions of EPI-patches are preserved along with the scene structural information, while also maintaining consistency between horizontal and vertical directions. However two questions remain to be answered: 1) whether the sharp edges will still be preserved in our final reconstructed image patch $\widehat{P_0}$ and 2) whether our model has enough representation power to reconstruct any image patches with arbitrary appearance?

By only looking at Eqns 2.12 and 2.13, we have not much to criticize in how well the sharp edges are preserved in $\widehat{P_0}$. Intuitively speaking, since we aim at preserving the sharp edges in EPI-patches, we hope the resulting image patch of P_0, a collection of corresponding rows/columns from EPI-patches, will also preserve the dominant edges. The experiments conducted in Fig 2.4 demonstrates this.

In terms of the second question, there are three components involved in our **patch-cube**s: image patch appearance, horizontal shift, and vertical shift. Achieving shift-invariance image representation is complicated due to the lack of reasonable priors on how the codes change with respect to the image shift. Modeling the behavior of image patches from different view points is even more challenging, since the amount of shift is determined by depth, and occlusions introduce even more complexity to the situation. Instead, our method essentially converts this problem to another space: image patch appearance, sparse coefficients $\{\alpha\}$ and $\{\beta\}$ for EPI-patches. Our optimization, Eqn 2.20, in spirit, is a linear model by reconstructing P_0 from horizontal and vertical directions separately. From Fig 2.4, we did notice, with both horizontal and vertical reconstructions, that neither is likely sufficient for reconstructing the original patch P_0, as it can be seen that the horizontal reconstruction has horizontal blur and the vertical reconstruction has vertical blur. However we can faithfully reconstruct the original image patch by merging the two, which suggests our model has sufficient representational power. Further experimental results will also be shown in Section 2.6. A logical next step is to consider multiplicative interactions between horizontal and vertical directions [31].

How to choose the size of t

EPI-patch dictionaries heavily rely on scene depth, since they are essentially capturing line segments with certain slope, where the slope range is determined by depth range. This

brings a requirement for the angular resolution of the light field camera – that the stacked EPI-images contain sufficient viewpoints that the EPI-patch pattern be easily determined.

The EPI-patch size t, see Fig 2.3, is empirically determined by *relative* depth, rather than the absolute depth of the scene. A common exercise is to use the *horopter* notion, which ensures that zero-disparity is positioned at the middle depth-of-field, which regularizes slope about a nominal zero. Here, the slope would be balanced about positive and negative, making line segments easier to extract. This also suggests that our method expects a canonical depth range, outside of which the line segments become difficult to extract within the EPI-patch. This will lead to our dictionary failing to cover all possible depths. However, with greater and greater angular resolution [14] (more view points), the slopes in an EPI-patch become easier to extract, therefore permitting greater range coverage.

The current **IBSC** model for single image analysis needs a fixed size filter, and is blind to geometry. It has been experimentally demonstrated in [45][44] that employing a multi-scale filter improves image denoising performance, however there is no principled way to choose an appropriate range of filters. On the other hand, we encode each image patch based on the geometry of its surround, its code will be automatically adjusted based on the scene geometry. For example, two similar appearing image patches will be encoded differently if one is a *true* edge located on an object and the other is a *false* edge located across a depth discontinuities. This provides a handle for incorporating geometrical information into our algorithms. That is, we will have the capability of manipulating or adapting the sparse coding model to handle different depth levels, or even occlusion. Also the simplicity of EPI-patches makes it possible to build a single dictionary which covers enough large depth range to ensure our learned features automatically adapted to the scene geometry.

To deal with very distant or very closer depth, we can adjust the *horopter* position to accommodate the far and near ranges. Essentially, we shift the EPI-patches toward that depth of interest so the slopes within that range are covered by the training EPI images. Although our trained dictionary can cover reasonably large depth of field, we need a mixture-of-model type of algorithm to systematically merge the model at different depth range, which will be left as future work.

2.5 Non-Local Mean image denoising in a light field

We have demonstrated that the **patch-cube** in a light field camera contains all necessary and complete information to describe an image patch. In this section, we will further demonstrate the value of the **patch-cube** by showing that it can provide a better similarity measure between image patches. More specifically, a simple and effective image denoising method – Non-Local Mean image denoising in a light field (NLM-LF) – is presented to demonstrate the power the EPI-based **patch-cube** can bring us.

Given an image patch in a reference image, NLM [18] searches for patches that are similar to it in the input image, see Eqn 2.4 and 2.5. The similarity measure, l_2 distance, is

[14] A joint angular/spatial super-resolution algorithm is proposed in Chapter 3.

susceptible to image noise and lighting condition, and inaccuracy in patch grouping lowers image denoising performance.

Our method differs from Buades et al. [18] by how the neighborhood is defined. Our neighborhood for a given pixel i is the horizontal and/or vertical **patch-cube**, that is,

$$\mathcal{N}(i) = \left\{i | i \in Q^{(h)}(i)\right\} \cup \left\{i | i \in Q^{(v)}(i)\right\}$$

To avoid text clutter, in the following we use only the horizontal **patch-cube**, and the **patch-cube**s at pixel i and j are respectively represented as:

$$Q^{(h)}(i) = \left[P^{(h)}_{-t}(i), \ldots, P^{(h)}_{0}(i), \ldots, P^{(h)}_{t}(i)\right] = \left[E^{(h)}_{-h}(i), \ldots, E^{(h)}_{0}(i), \ldots, E^{(h)}_{h}(i)\right]$$
$$Q^{(h)}(j) = \left[P^{(h)}_{-t}(j), \ldots, P^{(h)}_{0}(j), \ldots, P^{(h)}_{t}(j)\right] = \left[E^{(h)}_{-h}(j), \ldots, E^{(h)}_{0}(j), \ldots, E^{(h)}_{h}(j)\right]$$

Similar to Eqn 2.5, the linear weights between pixel i and j can be rewritten as:

$$w(i,j) = \frac{1}{Z(i)} exp\left\{\frac{1}{2t+1}\sum_{t'=-t}^{t}\frac{-||P^{(h)}_{t'}(i) - P^{(h)}_{t'}(j)||_2^2}{\sigma^2}\right\} \tag{2.36}$$

$$\sim \frac{1}{Z(i)} exp\left\{\frac{1}{2h+1}\sum_{h'=-h}^{h}\frac{-||E^{(h)}_{h'}(i) - E^{(h)}_{h'}(j)||_2^2}{\sigma^2}\right\} \tag{2.37}$$

where $Z(i)$ is the normalizing constant. Eqn 2.36 calculates the Euclidean distance using the local image patch in a **patch-cube**, while 2.37 is based on the EPI-patches. Essentially we want to re-adjust the patch similarity measure so that it is not only based on color and appearance, but also on scene geometry. Our motivations are as follows:

- If two patches around pixels i and j have similar color patterns and similar depth, then Eqn 2.36 acts as an averaging, not differing much from the conventional Non-Local Mean;
- If two patches have similar appearance, but different depth, their two patches cannot be differentiated using NLM. However, their EPI-patches will have quite different line patterns (slopes) which leads to reduced similarity. Larger depth differences usually suggest two patches are not from the same object;
- If two patches have similar appearance but one has *true* edges induced by texture differences on the object and the other has *false* edges induced by depth discontinuities or occlusion, the similarity will decrease;
- If two patches have similar appearance but one patch has an occlusion and the other does not, from the EPI-patches perspective, one will have broken lines, which is a strong sign that the two patches cannot be similar;

- Since observed patch color depends on lighting conditions, it will be sensitive to illumination variation and sensor noise; our similarity measure, by focusing on structural information, will be not be influenced by such noise;
- Conventional Non-Local Mean only measures pixel similarity between patches based on appearance or color, not scene structure, so will not perform as well where these elements are noisy.

To validate our algorithm, we performed the following experiments on the Stanford Light-field archive. Given 17×17 images, to each of which we have added *i.i.d.* Gaussian noise of $\mathcal{N}(0, \sigma^2)$, we compare the denoising effects for the central view, $(9, 9)$-th camera. The denoising comparison results are shown in Fig 2.8 ($\sigma = 35$) and 2.9 ($\sigma = 50$). [15] In both figures, the first and second column show the comparison of denoising results between [18] and ours. The third and fourth column further show the detailed comparison on a small region. From the figures, we can see significant improvement in image quality, especially at depth discontinuities and high-texture locations. For example, the pixel-based similarity measure used in [18] can not distinguish the helmet from the background (see the third-column of Fig 2.8-b), which causes helmet's edges in the denoised image to be blurred. However, this geometric information is evident in the setting of a **patch-cube**, see the clean helmets's contour in the recovered images using our method (the fourth-th column of Fig 2.8-b). More interestingly, our method performs much better in repetitive patterns (see the fourth column of Fig 2.8-b and 2.9-b).

All of these suggest considering scene geometric information can boost denoising performance. In particular, in area of high magnitude noise, the performance boost increases, suggesting that structure is as important as the appearance/intensity.

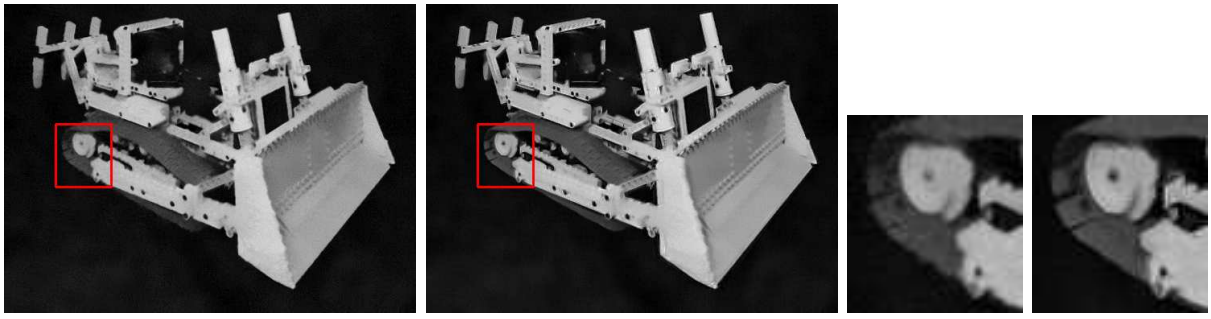
2.6 Experimental results

We test our new image patch model in the context of single image denoising in a light-field on the Stanford Light-field archive [5]. This dataset contains the light-field (17×17 views) data for 11 different scenes. The (9,9)-view of each scene is shown in Fig-2.10.

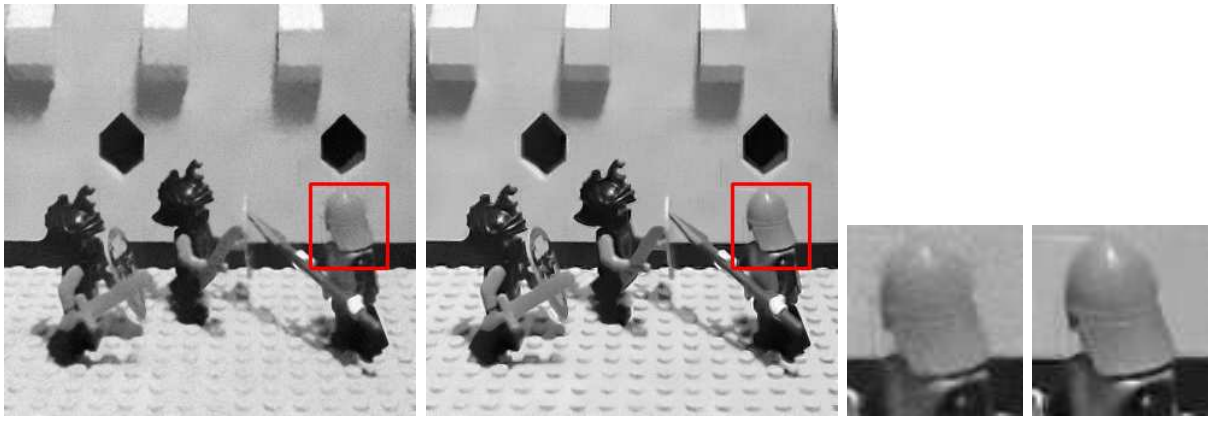
In this chapter, we inject the same magnitude of white noise on all light-field images, $\sigma = 25, 35, 50, 75$. Our goal is to denoise the central view image, $(9, 9)$-th camera, from all the noisy images. In order to better understand our algorithm, the following quantities are inspected:

- **I-H**: We denoise the image using only the horizontal **patch-cube**. The horizontal parameters $\{\alpha^0\}$ are initialized using the sparse coding model [52];
- **I-V**: We denoise the image using only the vertical **patch-cube**. The vertical parameters $\{\beta^0\}$ are initialized using the sparse coding model [52];

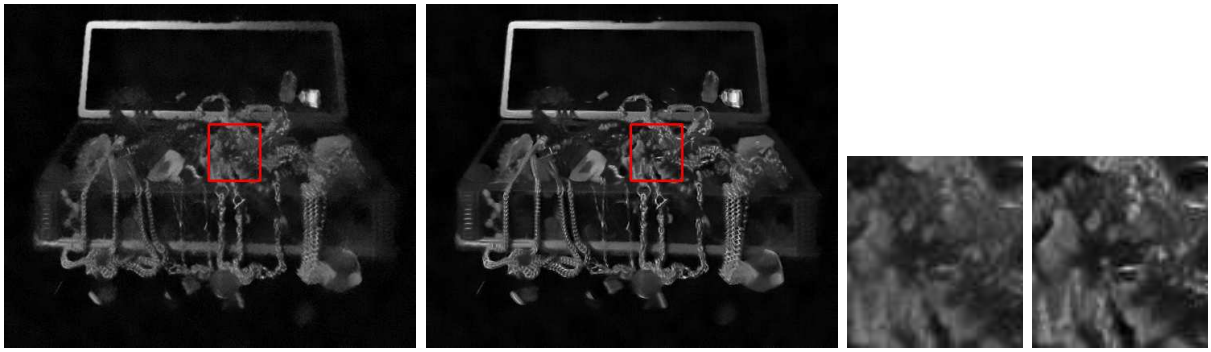
[15] We only use horizontal **patch-cube**.



(a) Lego Bulldozer

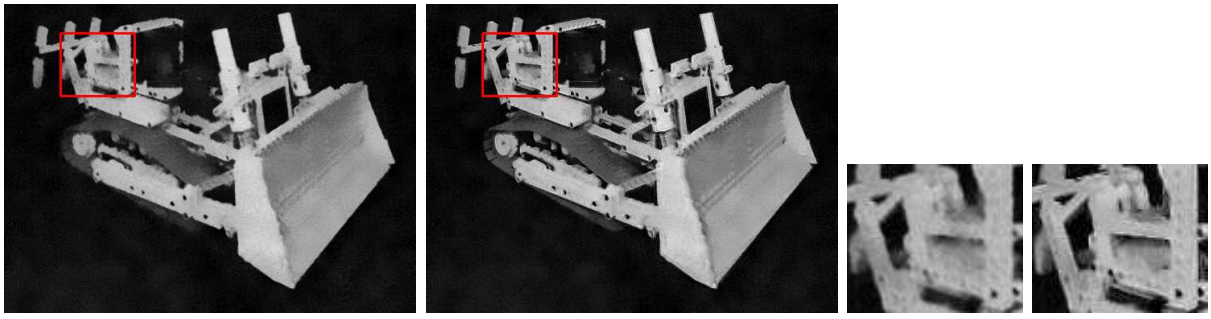


(b) Knights

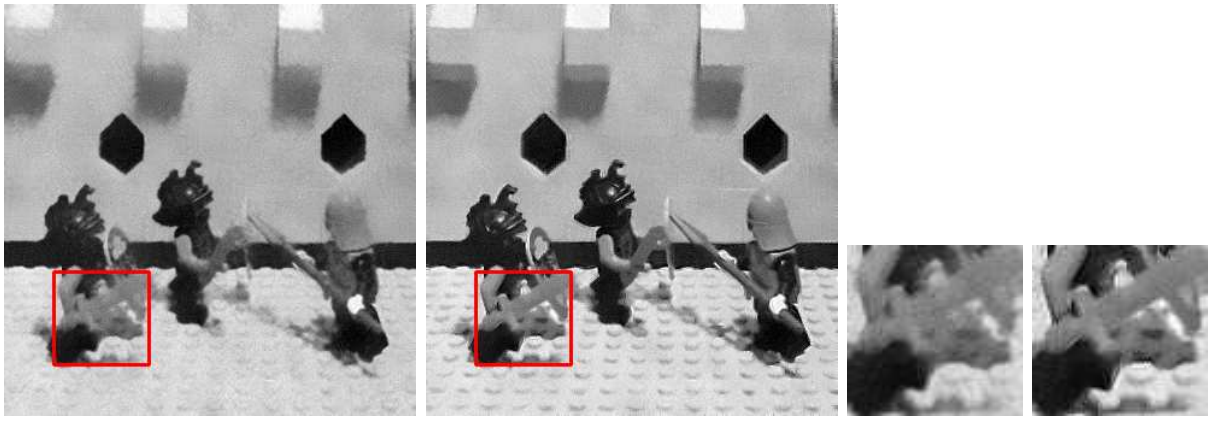


(c) Treasure

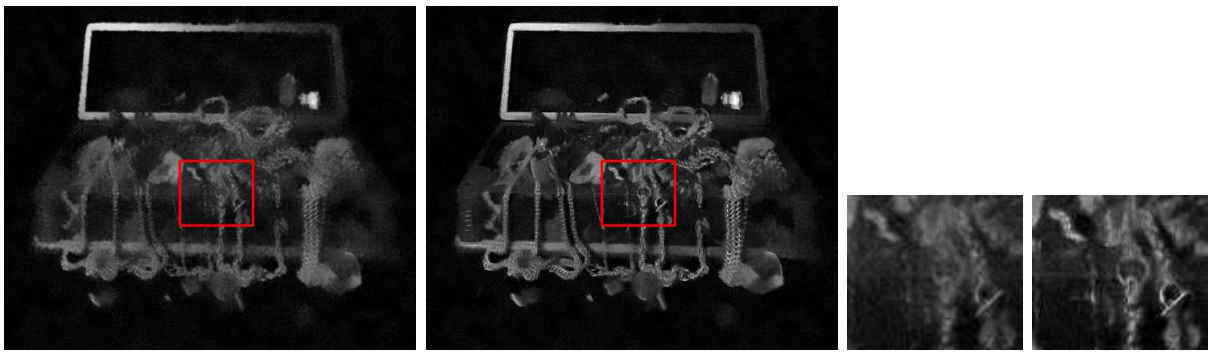
Figure 2.8: Denoising comparison with single-image NLM [18] ($\sigma = 35$): The 1st and 2nd column show the results using [18] and ours, respectively. The 3rd ([18]) and 4th (ours) columns show the detailed comparison on a small region.



(a) Lego Bulldozer



(b) Knights



(c) Treasure

Figure 2.9: Denoising comparison with single-image NLM [18] ($\sigma = 50$): The first and second column show the results using [18] and ours, respectively. The 3rd ([18]) and 4th (ours) columns show the detailed comparison on a small region.

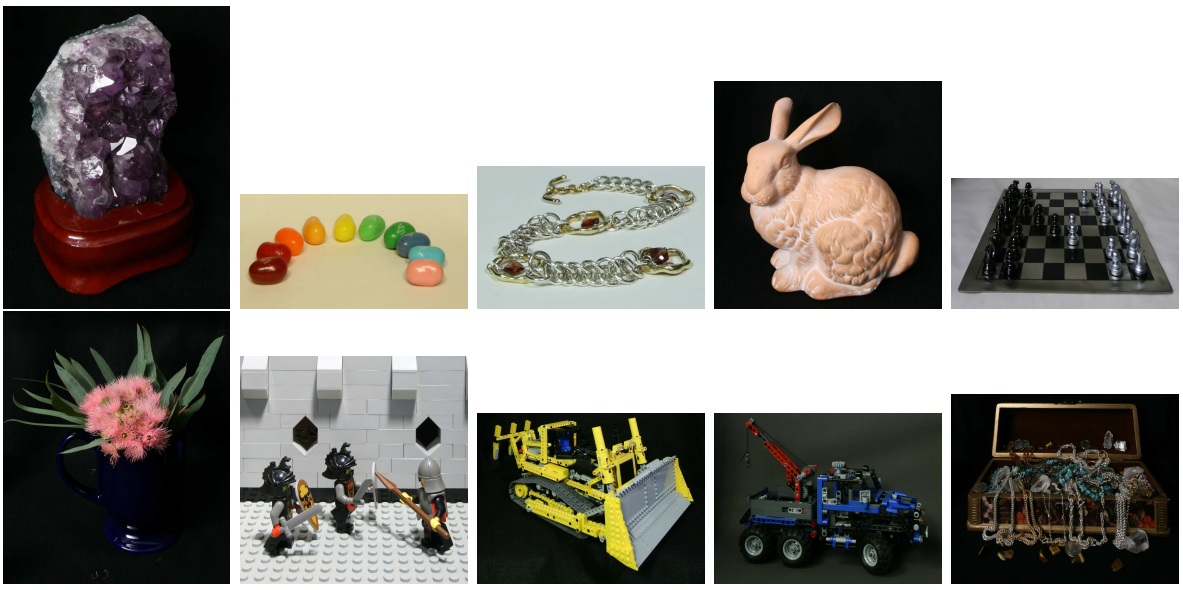


Figure 2.10: Sampled images from [5]: Amethyst, Beans, Bracelet, Bunny, Chess, Flowers, Knights, Lego Bulldozer, Lego Truck, and Treasure.

- **F-H**: We run Algorithm 3 five iterations, and the denoised image is calculated using only the horizontal optimized parameters $\{\alpha^*\}$
- **F-V**: We run Algorithm 3 five iterations, and the denoised image is calculated using only the vertical optimized parameters $\{\beta^*\}$
- **F-HV-1**: We run Algorithm 3 one iterations, and the denoised image is calculated using both the horizontal and vertical optimized parameters;
- **F-HV-5**: We run Algorithm 3 five iterations, and the denoised image is calculated using both the horizontal and vertical optimized parameters $\{\beta^*\}$;
- **I-H-W**,**I-V-W**, **F-V-W** and **F-V-W**: horizontal/vertical weights at initialization, and at optimal;
- **I-Err**,**F-Err**: image reconstruction error at initialization and optimal.

Detailed comparison of PSNR

The PSNR (Peak Signal Noise Ratio) for the different image datasets in the Stanford Lightfield archive are summarized in Table 2.1. From the table, we observe:

- Our algorithm, **F-HV-5**, performs significantly better than the **IBSC** model, for example [27], especially on the situations with high noise level;

Image/(σ)		PSNR						
		I-H	**I-V**	**F-H**	**F-V**	**F-HV-1**	**F-HV-5**	**[27]**
Amethyst	35	32.5457	32.6176	32.9831	33.2844	33.6184	**33.6922**	31.5822
	50	32.0152	32.0999	32.0699	32.3300	**32.7118**	32.5570	29.8224
	75	29.0311	29.0657	29.4447	29.4657	29.5163	**29.7509**	27.8152
bunny	35	33.2239	33.1330	34.1205	33.9643	34.0439	**34.5471**	33.8587
	50	32.9055	32.9188	33.2175	33.2089	33.4561	**33.5274**	31.8980
	75	29.4462	29.4039	30.0503	29.9184	29.8859	**30.2843**	29.4007
Chess	35	31.5215	31.7107	32.1313	32.6637	33.1296	**33.1994**	32.2640
	50	31.1993	31.3207	31.3159	31.7489	**32.2902**	32.0829	30.0100
	75	28.1304	28.1873	28.5679	28.6479	28.7438	**28.9727**	27.5294
Knights	35	29.5858	29.9062	30.3066	30.9681	**31.8470**	31.5461	31.1107
	50	29.0123	29.3707	29.2993	29.9107	**30.6443**	30.2827	28.9437
	75	26.7950	27.0862	27.1611	27.4653	27.6710	**27.7287**	26.6358
Bull	35	29.6552	29.3713	29.9075	30.0317	31.0879	30.6945	**31.4043**
	50	29.0487	28.8326	29.1096	29.2001	**30.1385**	29.7359	29.5011
	75	26.7839	26.6038	26.9437	26.8812	27.2656	27.2483	**27.2858**
Truck	35	32.0428	32.3664	32.6501	33.1682	33.3228	**33.5574**	32.1797
	50	31.7155	32.0086	31.8458	32.2828	**32.5528**	32.4658	29.9994
	75	28.4709	28.6106	28.9845	29.0686	29.0275	**29.3419**	27.5779
Treasure	35	29.2864	29.5291	28.9691	29.9115	**30.9015**	30.0349	30.1758
	50	28.8305	28.9955	28.4271	29.2113	**30.0748**	29.3117	28.1133
	75	26.5061	26.6679	26.4014	26.7774	**27.1209**	26.8780	25.6871
Flowers	35	32.0056	32.0207	32.1882	32.4914	**32.8270**	32.7735	31.7788
	50	/	/	/	/	/	/	/
	75	28.6283	28.6886	28.9206	28.9926	29.0424	**29.1971**	27.6488

Table 2.1: Comparison of PSNR at each stage

- On the most of datasets and noise levels, **I-H** or **I-V** perform similar, both better that those using [27]. That is, denoising the image either using horizontal or vertical **patch-cube** provides improvement;
- **F-H** and **F-V** perform better than their counterparts, **I-H** and **I-V**, respectively;
- **F-HV-5** performs slightly better than **F-HV-1**, especially for the situations of higher noise level. This might be due to an overly strong initialization of the sparse coding model on EPI-patches, which leads the highly non-linear optimization in Eqn 2.23 getting stuck at local minima. In the future, we will investigate different initialization strategies.

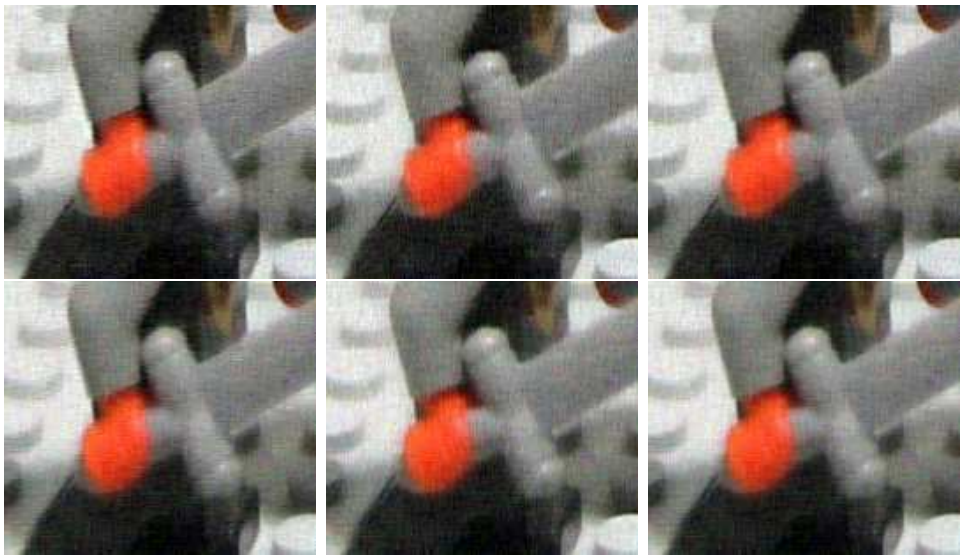


Figure 2.11: Detailed comparison in a small region for Fig 2.14 ($\sigma = 75$): From top to bottom, left to right: **I-H**, **I-V**, **F-HV-1**, **F-H**, **F-V**, **F-HV-5**

Denoising performance at different stages

In this section, we visually evaluate the performance of our algorithm at different stages using the image dataset **Knight**. The results are shown in Fig 2.12, Fig 2.13 and Fig 2.14, for noise levels $\sigma = 35, 50, 75$, respectively. To get a detailed comparison, we also choose a small window in Fig 2.14 ($\sigma = 75$) and visualize the denoised results from different stages, see Fig 2.11. As we expected, 1) at the initialization stage, the denoised results using **patch-cube** along only one direction, either horizontal (**I-H**) or vertical (**I-V**), are less satisfying. For example, **I-H**, focusing on the horizontal part of a light-field, tends to blur along the vertical edges; 2) Once we consider both directions, even at initialization (**F-HV-1**), these blurring effects diminish. After the optimization, both horizontal and vertical denoising performance improved. For example, **F-H** vs **I-H**, and **F-V** vs **I-V** although we still use only one direction. One thing worth mentioning: the sharp edges can be well preserved both at initialization and after optimization. The latter seems to slightly smooth the edges. For example, **F-HV-5** is less sharper than **F-HV-1**. However the former is more visually appealing without losing the sharp edges, especially at depth discontinuities.

Our algorithm dynamically adjusts the weights based on scene geometry, both horizontally and vertically, during the optimization. More weights will be put on the highly texture-rich regions and depth discontinuities. For example higher weights are observed horizontally and vertically in **I-W** and **F-W**, respectively. These are places indicating the scene geometry change, for example occlusion, and then our algorithm will focus on those places. Comparing **F-HV-5** and **F-HV-1**, the weights are much reduced after the optimization. Also the former performs better at the background texture-less regions. This type of noise

is easily removed once the geometric information is considered. Also comparing **I-Err** and **F-Err**, the reconstruction error is reduced overall, the depth discontinuities drop the most. This suggests that our algorithm models these places particularly well by exploiting the scene geometric structure.

Across all noise levels, improvements increases with noise. This is one of the advantages of our model: geometric information is less affected by image noise.

Image denoising comparison

In this section, we compare our results with the method in [27] at two different noise level: $\sigma = 35$ and $\sigma = 75$. See Fig 2.15 for the overall comparison on the **Knight** image dataset [5]. In Fig 2.16, we compare in detail by inspecting a small region for all image dataset: in each sub-figure, 1) the interested region is shown in the top-row, 2) the first and second column in the second-row show the denoising comparison at $\sigma = 35$, and 3) the third and fourth column in the second-row show the denoising comparison at noise level of $\sigma = 75$.

From the figures, we can see that our algorithm performs much better in locations of depth discontinuity. Also [27] blurs those high-contrast edges. This is because the image patches there contain change due to color/texture variation, and a single sparse coding method might be not adequate for faithful reconstruction. Also on texture-less regions, [27] seems not very effective, retaining too much noise. Further, the sharper edges seems more artificial than the results we produced. Lastly, large improvements observed comparing to [27], especially on higher noise level.

2.7 Conclusion and future work

In this chapter, a new image patch model based on EPI-image analysis has been proposed. The model considers not only the appearance or color of the interested image patch, but also the scene geometric information surrounding it. The major edges on both horizontal and vertical EPI-patches are well preserved, and the consistency from horizontal and vertical reconstruction is ensured. The former empowers our representation to respect scene geometry. Also, we use a very simple strategy to improve the conventional NLM algorithm for denoising. This viewpoint dependent representation of an image patch, **patch-cube**, brings new opportunities to machine learning and computer vision research. In the next chapter, we will extend this idea to joint denoising and joint angular/spatial super-resolution applications in a light-field camera.

Due to the special structure of the **patch-cube**, a future research effort might consider this in a more systematic way. For example, the EPI-patches in a **patch-cube** can be modeled using the manifold learning approach, and achieving viewpoint-invariant image representation arises as an interesting topic.

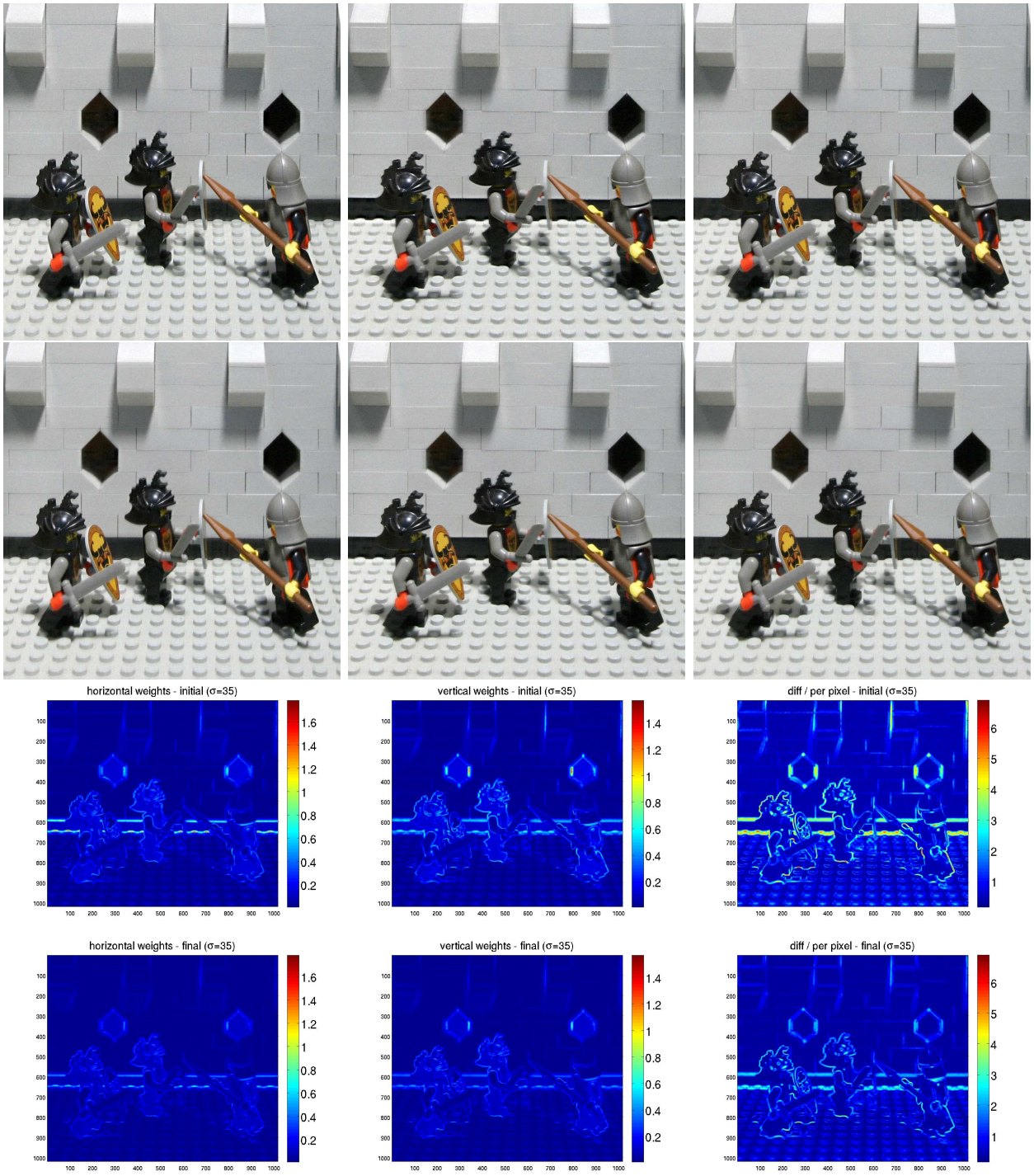


Figure 2.12: Denoise performance for $\sigma = 35$: From top to bottom, left to right: **I-H**, **I-V**, **F-HV-1**, **F-H**, **F-V**, **F-HV-5**, **I-H-W**, **I-V-W**, **I-Err**, **F-H-W**, **F-V-W** and **F-Err**

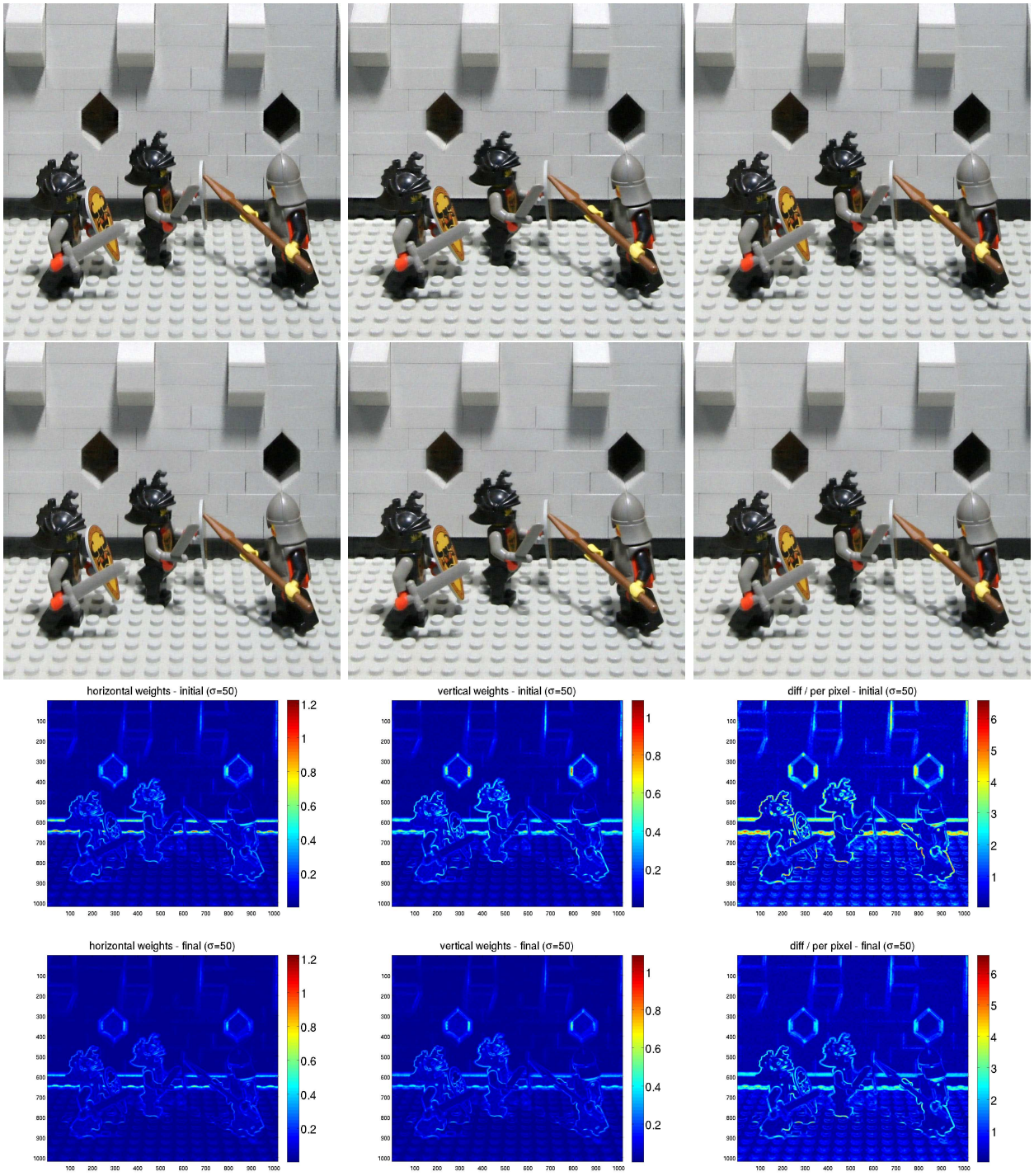


Figure 2.13: Denoise performance for $\sigma = 50$: From top to bottom, left to right: **I-H**, **I-V**, **F-HV-1**, **F-H**, **F-V**, **F-HV-5**, **I-H-W**, **I-V-W**, **I-Err**, **F-H-W**, **F-V-W** and **F-Err**

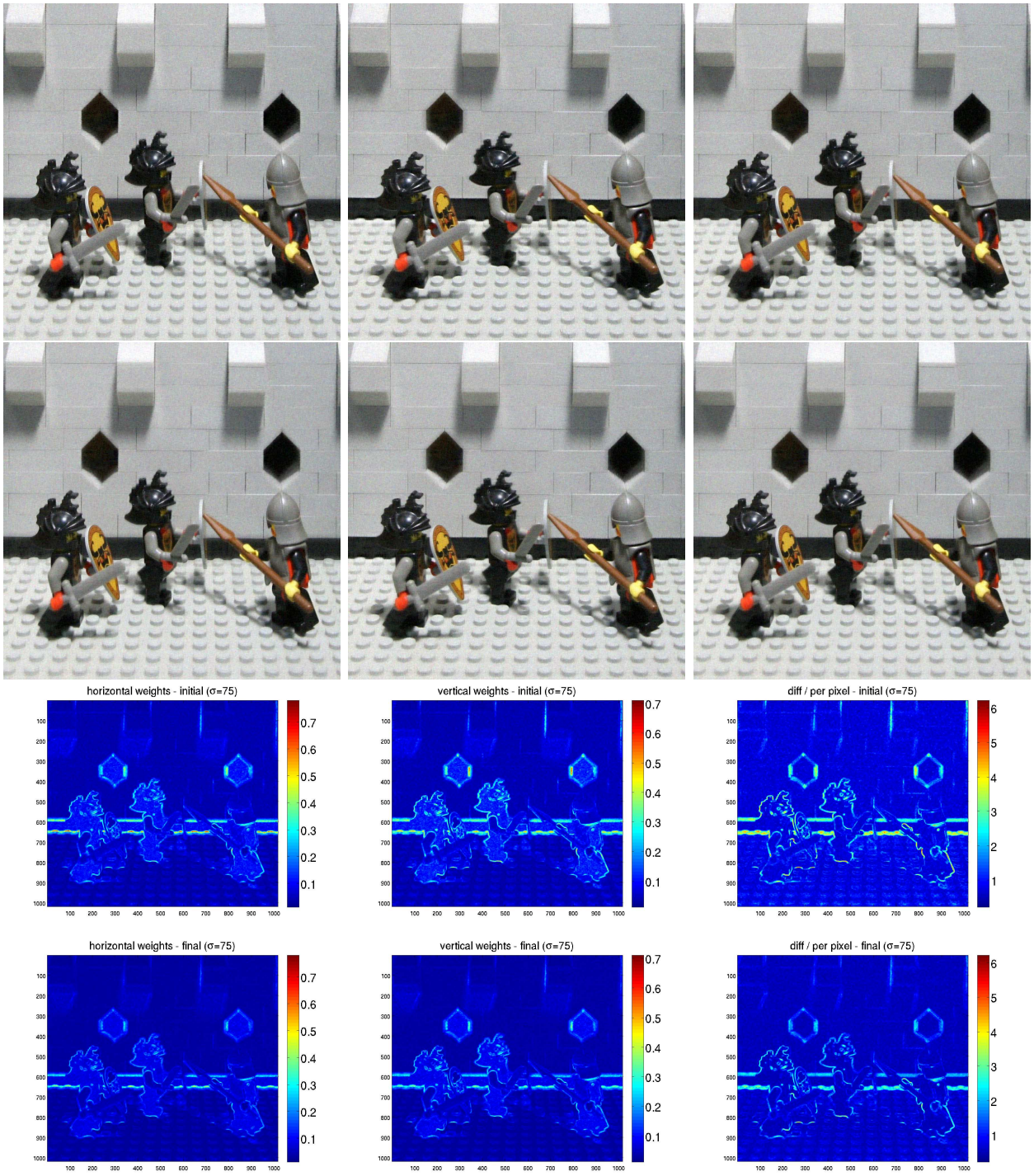
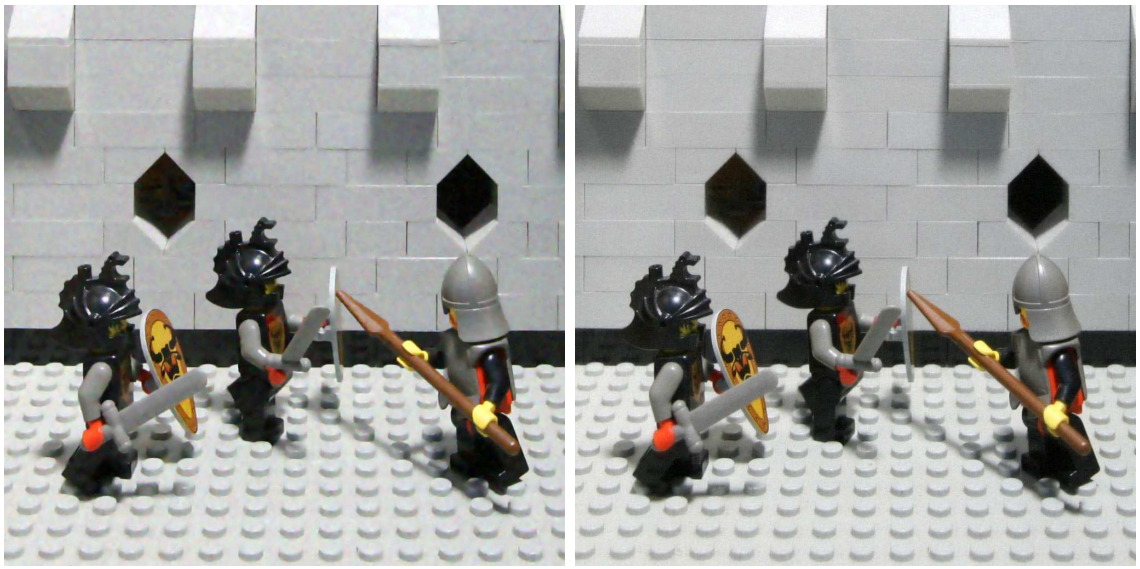
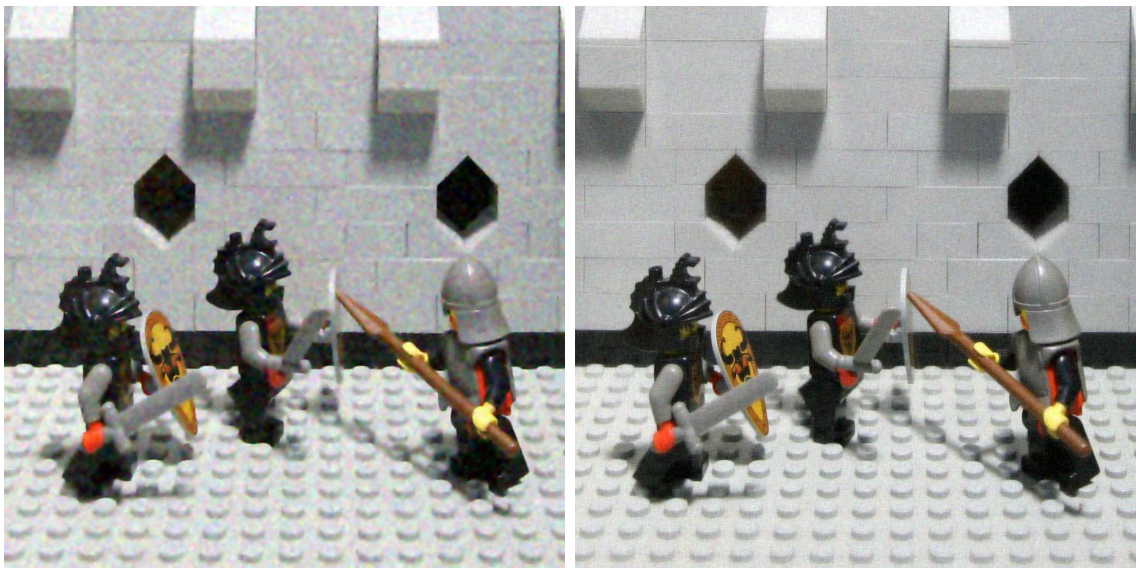


Figure 2.14: Denoise performance for $\sigma = 75$: From top to bottom, left to right: **I-H**, **I-V**, **F-HV-1**, **F-H**, **F-V**, **F-HV-5**, **I-H-W**, **I-V-W**, **I-Err**, **F-H-W**, **F-V-W** and **F-Err**

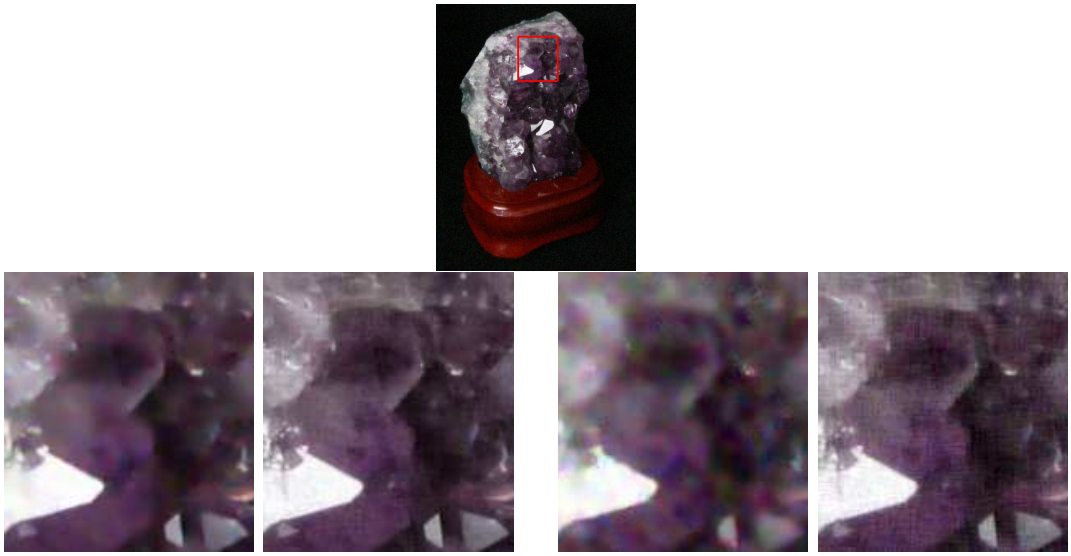


(a) Knight ($\sigma = 35$)

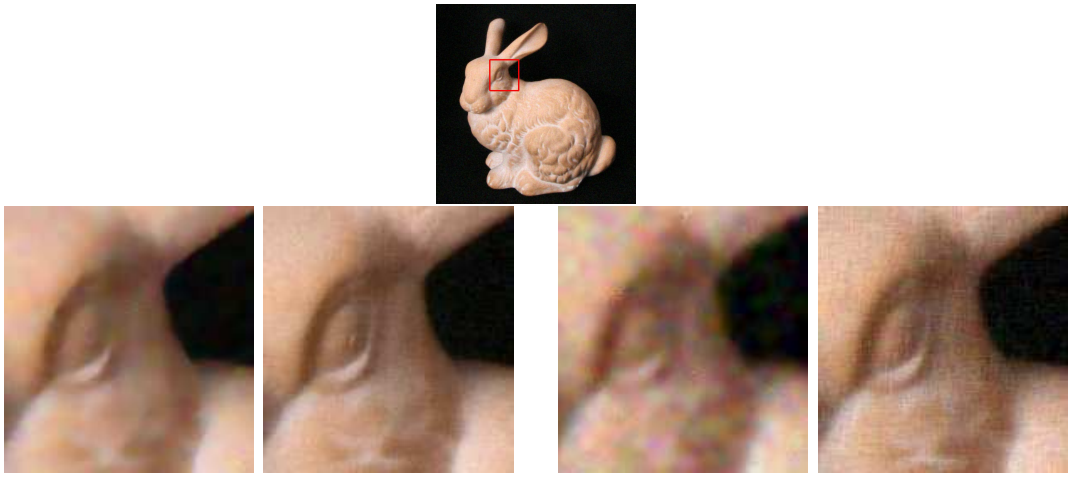


(b) Knight ($\sigma = 75$)

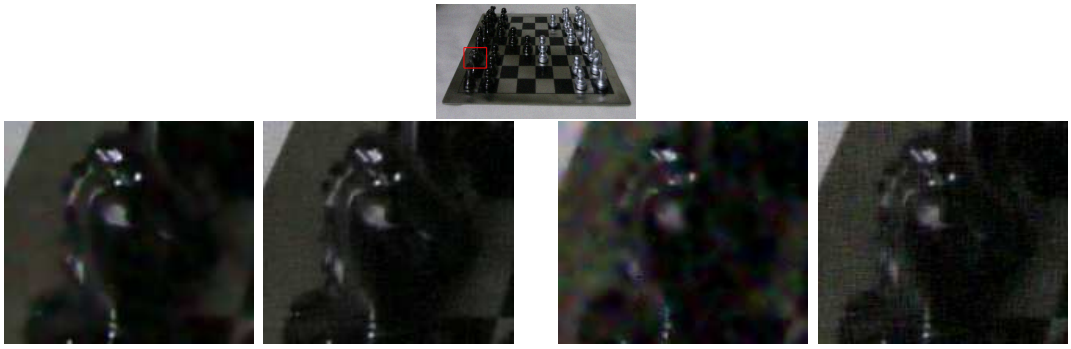
Figure 2.15: Denoising performance comparison for $\sigma = 35/75$ on **Knights** image dataset. Left: [27], Right: ours



(a) Amethyst

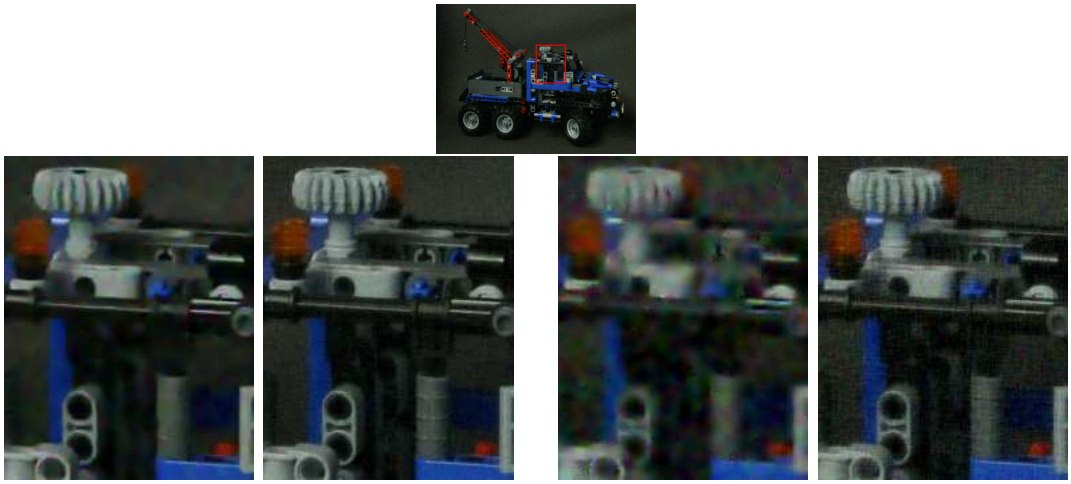


(b) Bunny

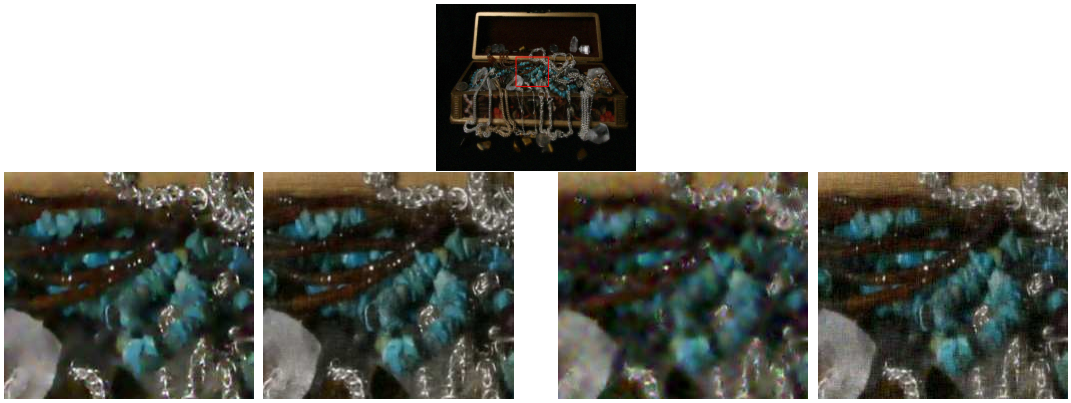


(c) Chess

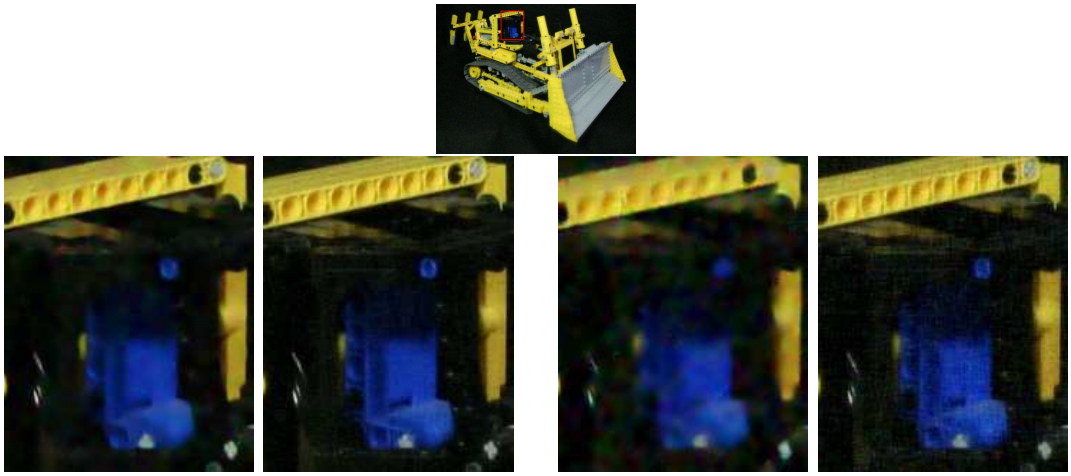
Figure 2.16: Detailed denoising performance comparison for $\sigma = 35/75$ on different image dataset. See the text for the details.



(d) Lego Truck



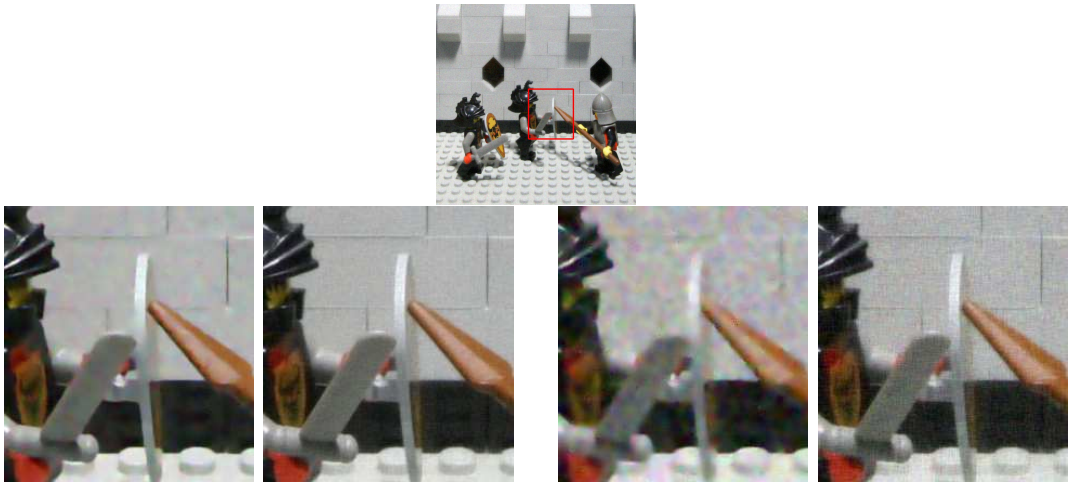
(e) Treasure



(f) Lego Bulldozer

Figure 2.16: (continued) Detailed denoising performance comparison for $\sigma = 35/75$ on different image dataset. See the text for the details.

(g) Flowers



(h) Knights

Figure 2.16: (continued)Detailed denoising performance comparison for $\sigma = 35/75$ on different image dataset. See the text for the details.

Chapter 3

Joint image denoising and spatial/angular super-resolution in a light field

In Chapter 2 we detailed our patch modeling in a light-field, by only considering the *cross*-structure in a 2D camera array for a given image patch. For single image denoising, it achieves better performance. In this chapter, we generalize this idea to leverage the full 2D set of viewpoints. Two different applications are presented to demonstrate the superior performance of our methods: joint image denoising and spatial/angular super-resolution in a light field.

3.1 Introduction

Image denoising and super-resolution are important tasks in the analysis of digital images. Much research has addressed these issues to date, with most being focused on a single image, for example denoising [18][27][11], and super-resolution [78][36]. The common theme is to frame the task as an optimization problem, seeking an improved image which is as close as possible to the noisy image under some regularization constraints, such as sparsity [27] or total variation [11]. These methods have been especially successful in preserving the sharpness of edges in the image. The generic framework, however, fails to perform well in the presence of high noise, and is independent of scene structure. Light field data presents an opportunity to move beyond these limitations.

There are many sources of image contamination and poor representation during digital capture, and these degrade the ability to process and properly assess content. Light-field data, in particular, has issues with noise, since current acquisition tends to use a single sensor with overlaid micro lenses, forcing pixel sizes to be small and sampling to be rather sparse. Also, the noise characteristics for each micro-lens or multi-camera are different. It is likely that one or more images in the whole 2D image sets have been corrupted in some

manner by noise. Using the remaining "good" images to correct the noisy one(s) is an economical solution. In this chapter, we are solving the extreme case where all of the images are potentially corrupted by some sort of noise.

On the other hand, light-field cameras usually have to balance the trade-off between spatial and angular resolution due to the fact that total sensor resolution is limited. The angular resolution is critical to EPI-based image analysis. The larger the angular resolution, the more viewpoints acquired of the scene, the more accurate the scene geometry that can be extracted through EPI-image analysis (see Chapter 2 for the detailed discussion). This will not only benefit depth retrieval from a light field, but also the image analysis based on the EPI-patches (see Chapter 2). However the angular resolution is usually limited by the camera physical manufacturing considerations and the spatial resolution of each camera. Our goal is to effectively interpolate novel viewpoints over the 2D grid, while still respecting the geometric constraints embedded in a light field.

In Chapter 2, a new image patch modeling method is presented to better incorporate scene geometric information, and to be less affected by image noise, lighting conditions, and so on. In this chapter, we consider beyond the *cross*-structure used in Chapter 2, and take full advantage of all of the viewpoints in a 2D camera array. The aim of this chapter is to leverage the rich information embedded in the EPIs and perform joint image denoising and angular/spatial image super-resolution tasks without explicitly recovering scene depth. More specifically, we focus more on the scene geometry by forcing the reconstructed EPI-images, after denoising or super-resolving, to maintain the characteristic patterns of their true counterparts.

We organize this chapter as follows: in Section 3.2, we describe the current state-of-the-art in image denoising and super-resolution computation. In Section 3.3 we present the two stages of our joint image denoising algorithm, and present a spatial/angular super-resolution algorithm in Section 3.4. The experimental results, conclusions and future work are given in Section 3.5, 3.6 respectively.

3.2 Previous work

There are not many works on joint image denoising and super-resolution [86][40][47]. Most recent efforts have been focused on single image denoising and super-resolution. Single image denoising algorithms usually model a noised image using the following generative process:

$$u(i) = v(i) + \epsilon \tag{3.1}$$

where $u(i)$ is the noisy/observed pixel, $v(i)$ is the cleaned value, and ϵ is the noise perturbation at pixel i, which are *i.i.d.* Gaussian noise with zero mean and variance σ^2.

One line of work, [27][52] is to model each image patch independently based on the sparseness principle [52], meaning each image patch can be modeled as a sparse linear combination of an over-complete dictionary. The reconstructed image patches from the sparse codes can effectively preserve the sharp edges within that patch and also suppress image noise. Instead

of sparseness constraints for each image patch, the Total Variation (TV) based method directly derives the cleaned image $\tilde{v}$ from u by constraining the problem using Total Variation [11]. Another line of work is related to the similarity between image patches, for example [18][24]. Bi-lateral filter [66] uses isotropic filters which depend on not only spatial distance but intensity difference to avoid blurring edge sharpness.

For super-resolution tasks, many recent works solve for single image super-resolution by learning a non-linear mapping from a low-resolution image patch to a high-resolution one. Yang et al. [79] assume the low-resolution and high-resolution local image patches share the same sparse coefficients, while the latter is over a high-resolution dictionary. [36] models this mapping as a kernel regression model. Both of these models are trained on a collection of (low-resolution, high-resolution) image-patch pairs collected beforehand.

In the context of multiple images, [86] proposes an image denoising algorithm in a multi-view camera setup, merging multiple image patches in a general setting by finding the correspondences across them. Wanner et al. [71][72] solve the angular and spatial super-resolution problem in a light-field using variational optimization. Levin at el. [40] model the EPI structure, or slope field, of a light-field using Bayesian analysis. Mitra [47] model the light-field patches using a Gaussian Mixture Models (GMM) conditioned on the particular depth. However, all approaches require depth information to be attained beforehand, which is time consuming and error-prone.

3.3 Two-stage joint image denoising

Our set up is as follows: Given a light field with $N \times M$ cameras, $\{I^{rc}\}$, each of which has size of $S \times T$ and has been contaminated by some type of noise, we want to get the cleaned version of all images in this light-field.

In order to denoise all images in a light-field, we could naively denoise each image I^{rc} separately using the method presented in Chapter 2. More specifically, I_{rc} can be regarded as a collection of image patches, $\{P^{ij}_{rc}\}$, representing an image patch P^{ij} in the (r, c)-th camera, each of which is associated with a horizontal and vertical **patch-cube** and can be denoised by our method in Chapter 2 to get the cleaned patch. The final denoised image $\widehat{I_{rc}}$ will be acquired by spatially averaging all of the denoised image patches. However this process is computationally intensive, and also does not consider all possible viewpoints. In the following, a simple method is proposed to overcome these shortcomings.

3.3.1 Our method

Here, a simplified two-stage denoising algorithm is presented for denoising all images in a light field. A sketch of the algorithm is shown in Figure 3.1. Our algorithm proceeds by performing horizontal denoising, $(a) \rightarrow (b)$, denoted as $\mathbf{hori}_1$, followed by vertical denoising, from $(b) \rightarrow (c)$, denoted as $\mathbf{vert}_1$:

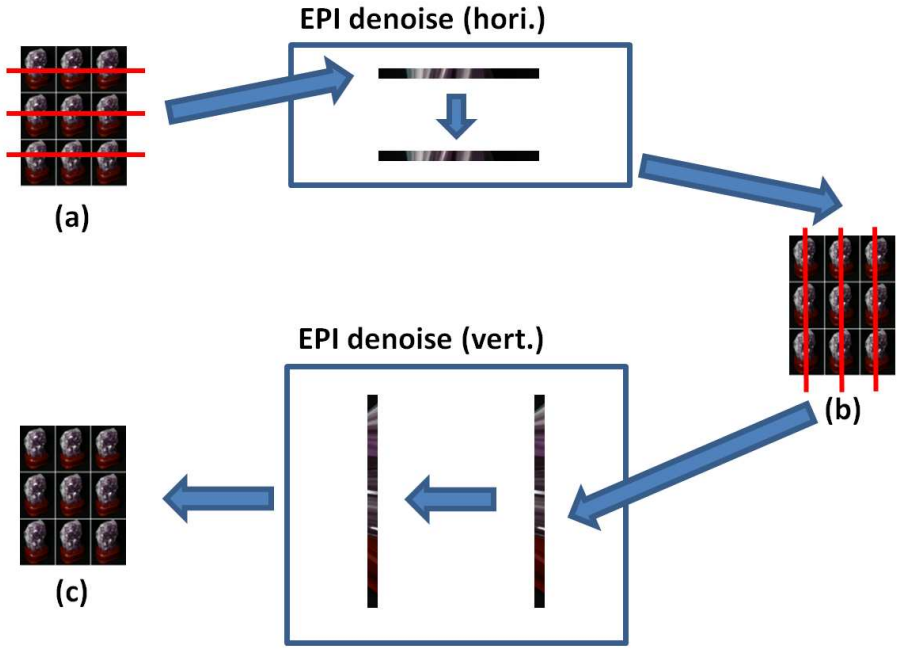


Figure 3.1: Joint image denoising diagram

- **hori$_1$**: Starting from all noisy images, we calculate the intermediate denoised images $\{\widehat{I_{rc}^{(h)}}\}$ by using only one row of cameras. For each row of the image, horizontal EPI image (where the EPI-patches are extracted from) is formalized and denoised as a regular image using the method in [27]. [1] The horizontally denoised images can be reassembled from these denoised EPI-patches.

- **vert$_1$**: Instead of using the original noisy images $\{I_{rc}\}$ as inputs, we conduct vertical image denoising using the intermediate denoised images $\{\widehat{I_{rc}^{(h)}}\}$ as inputs. More specifically, the vertical EPI-images are constructed from the horizontally (partially) denoised images in a light field, and then denoised.

This above procedure can be thought of as one-step iteration of the more general algorithm Algorithm 3 presented in Chapter 2. Let's look at the behavior for a given arbitrary image patch P_0. The horizontal denoised patch can be calculated by only considering the

[1] Any off-the-shelf single image denoising algorithm can be used.

horizontal **patch-cube** $Q^{(h)}$ as:

$$\widehat{P_0^{(h)}} = \begin{bmatrix} \left(D_0^{(h)}\alpha_1\right)^T \\ \left(D_0^{(h)}\alpha_2\right)^T \\ \vdots \\ \left(D_0^{(h)}\alpha_n\right)^T \end{bmatrix}$$

where $\{\alpha_i\}$ are optimized by solving:

$$\alpha_i^* = \underset{\alpha_i}{\operatorname{argmin}} \quad \left|\left| vec^{(h)}\left[E_i^{(h)}\right] - D^{(h)}\alpha_i \right|\right|_2^2 + \lambda|\alpha_i|_1 \tag{3.2}$$

Similarly, the vertical denoising step will only consider the vertical **patch-cube**. However, instead of constructing the vertical EPI-patches using the original noised images, we use the already horizontally denoised images $\{\widehat{I_{rc}^{(h)}}\}$. That is,

$$\widehat{P_0^{(final)}} = \begin{bmatrix} D_0^{(v)}\beta_1 & D_0^{(v)}\beta_2 & \dots & D_0^{(v)}\beta_n \end{bmatrix}$$

$\{\beta_j\}$ are solved by minimizing:

$$\beta_j^* = \underset{\beta_j}{\operatorname{argmin}} \quad \left|\left| vec^{(v)}[\widehat{E_j^{(v)}}] - D^{(v)}\beta_j \right|\right|_2^2 + \eta|\beta_j|_1 \tag{3.3}$$

where $\widehat{E_j^{(v)}}$ represents the vertical EPI-patches extracted from the partially denoised images $\{\widehat{I_{rc}}\}$, instead of the noised images $\{I_{rc}\}$. Notice the difference between these equations and those in Chapter 2. Also, note that our final denoised image is not simply the average the horizontally and vertically denoised images.

The on-line nature of this updating rule in our simplified algorithm usually leads to faster convergence. Further, since the vertical denoising is built upon the horizontally (partially) denoised images, we essentially implicitly consider all of the relevant patches for P_0, not just the *cross*-structure used in Chapter 2. More specifically, given the image patch P_0, all of the corresponding patches from the 2D grid of viewpoints will contribute to its modeling. The appearance or color of these patches might change due to viewpoint change, however EPI-patches on each row and each column, or even diagonally, will reveal part of the scene geometric structure surrounding P_0. This highly redundant information, or geometric constraint, makes our image patch modeling more stable, and less affected by image noise and lighting conditions.

The algorithm is summarized in Algorithm 4. It steps through the horizontal denoising $\mathbf{hori}_1$, followed by the vertical denoising $\mathbf{vert}_1$. Similarly, we can also run the optional steps by denoising vertically $\mathbf{vert}_2$ first and then horizontally $\mathbf{hori}_2$. The final denoised images will

be the average of the two steps. The effects of this extra step will be examined experimentally in Section 3.5. Also note the sparsity constraint Lagrange multiplier in the above two optimizations, see Eqn 3.2 and Eqn 3.3. Since Eqn 3.3 uses the partially denoised image, we expect less sparsity in the vertical sparse coefficients $\{\beta_j\}$ to be sufficient. Therefore, choosing the appropriate Lagrange multiplier is experimentally important.

Algorithm 4 Joint image denoising

Require: All noisy light-field images $\{I_{(i,j)}\}$, where $i = 1, \ldots, N$, $j = 1, \ldots, M$;
Ensure: All denoised images $\{\tilde{I}_{(i,j)}\}$

```
//horizontal denoising (Hori_1):
for each row (i) of the 2D camera array do
    for each row (y) of the images do
        Form horizontal EPI E^(h)_{i,y} using {I_{i,1:M}}
        Denoise E^(h)_{i,y} to get cleaned version Ê^(h)_{i,y}
    end for
    Construct horizontal denoised images for i-th row of 2D array {Î_{i,1:M}}.
end for

//vertical denoising (Vert_1):
for each column (j) of the 2D camera array do
    for each column (x) of the images do
        Form vertical EPI Ê^(v)_{j,x} using horizontally denoised images {Î_{1:N,j}};
        Denoise Ê^(v)_{j,x} to get cleaned version Ẽ^(v)_{j,x};
    end for
    Construct the final denoised images for j-th column of the camera array: Ĩ_{1:N,j}.
end for
```

In the following, we will take a closer look at the joint denoising algorithm.

3.3.2 EPI image denoising effect

The core part is the denoising on EPI-patches instead of the regular image patches, where scene geometric information has been automatically encoded. During this process, we expect the EPI-image properties will be preserved as much as possible after our denoising.

To examine how the EPI-image denoising affects the reconstructed image, we run the following experiments on **Amethyst** dataset from Stanford Light-field archive [5] (see Section 2.6 in Chapter 2 for detail), shown in Fig 3.2. We construct the horizontal EPI-image from **Amethyst** dataset at $y = 210$, both from the original images (see Fig 3.2-(a)) and from the noisy images (see Fig 3.2-(b)). Fig 3.2-(c) and Fig 3.2-(d) show the denoised EPI-images after ($\mathbf{Hori}_1$) and ($\mathbf{Vert}_1$), respectively. Fig 3.2-(e) shows our final results. As a comparison,

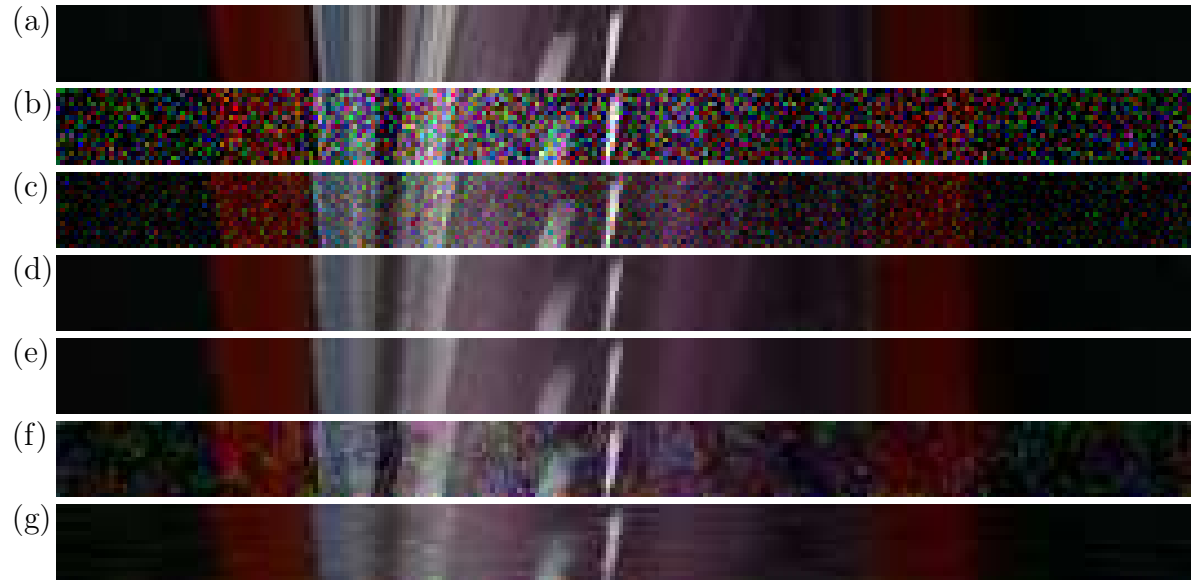


Figure 3.2: EPI image denoising effects in a light-field. (a) Noise-free EPI; (b) noisy EPI; (c) denoised by $Hori_1$; (d) denoised by $Hori_1 + Vert_1$; (e) final denoised EPI; (f) denoised by [11]; (g) denoised by [27]

we also test the EPI-image denoising effect using [11] and [27] as follows: We run [11] or [27] to denoise the noisy images for all of the 2D images separately and then extract the corresponding EPI-image from the denoised images. The results are shown in Fig 3.2-(f) and Fig 3.2-(g), respectively. From the above figures, we can see that:

- Our algorithm not only effectively preserves the dominant edges in EPI-images, but also is effective in removing noise in texture-less regions;
- Both [11] and [27] perform poorly, the structural information embedded in the EPI-images has been lost. This suggests that considering each image patch individually without knowing the scene geometry cannot preserve the scene geometry;
- In our algorithm, the horizontal denoising step, $\mathbf{Hori}_1$, is not sufficient to remove all of the noise, see Fig 3.2-(c). However combining with the vertical denoising step, $\mathbf{Vert}_1$, it is more powerful;
- More interestingly, [27] blurs the EPI-images after denoising, which is what we have observed in Chapter 2.

Since we are implicitly leveraging the scene structural information, our algorithm is also more robust to noise than others. More experimental results will be given in Section 3.5.

3.4 Spatial/Angular super-resolution in a light field

In Section 3.3, a new algorithm was presented to jointly denoise all images in a 2D grid camera array. We have shown it is a simplified version of the more general image patch modeling method presented in Chapter 2. In this section, we will apply the same idea to super-resolve a light-field both angularly and spatially.

Out setup is as follows: Given $N \times M$ images, each having size $S \times T$, we want to get $(3N) \times (3M)$ images, each of size $(3S) \times (3T)$. That is, we do viewpoint interpolation and triple the number of views. The spatial resolution of each image also will be increased.

3.4.1 Our algorithm

The commonly used methods for single image super-resolution computation model each image patch independently, without knowing the scene geometric information. We argue that this strategy is insufficient to maintain scene geometry. Instead, due to the dual view of the **patch-cube**, we focus more on the geometry-related EPI-patches, which automatically encode this scene geometry, including image-patch depth, occlusion, and so on. The resulting representation for an image patch will better preserve scene geometry. Slightly different from our jointly denoising algorithm, the key insight here is that the special structure of an EPI-patch should be preserved after super-resolution.

Our algorithm diagram is shown in Fig 3.3, it proceeds as follows, and also presented as in Algorithm 5:

- Super-resolving horizontally ($\mathbf{Hori}_1$): Starting from the original low-resolution images on each row of cameras, we build the intermediate partially super-resolved images, each of which has size $S \times 3T$. More specifically, we treat each horizontal EPI-image (of size $M \times T$) as a regular image and super-resolve it using any off-the-shelf single image super-resolution algorithm. The intermediate partially super-resolved images are reassembled from all of the above super-resolved EPI-images (of size $3M \times 3T$);

- Super-resolving vertically ($\mathbf{Vert}_1$): From the partially super-resolved images generated from the first step, we collect all EPI-images (of size $S \times N$) for each column of cameras and super-resolve them into size $3S \times 3N$, from which the final super-resolved images (of size $3S \times 3T$) are formed by reassembling the resulting EPI-images;

- (Optional) Post-processing to get rid of ghost imaging along the dominated edges with very strong intensity changes. This step can be achieved by running some smoothing filter or Markov Random Field model.

During ($\mathbf{Hori}_1$), only the horizontal spatial resolution is enlarged $3\times$, with the vertical viewpoints being interpolated implicitly. Specifically, a horizontal EPI-image with size $M \times T$ in the low-res light-field will be super-resolved into $3M \times 3T$. Since the EPI-constraints, both the connected and broken line segments, only characterize the property of scene geometry

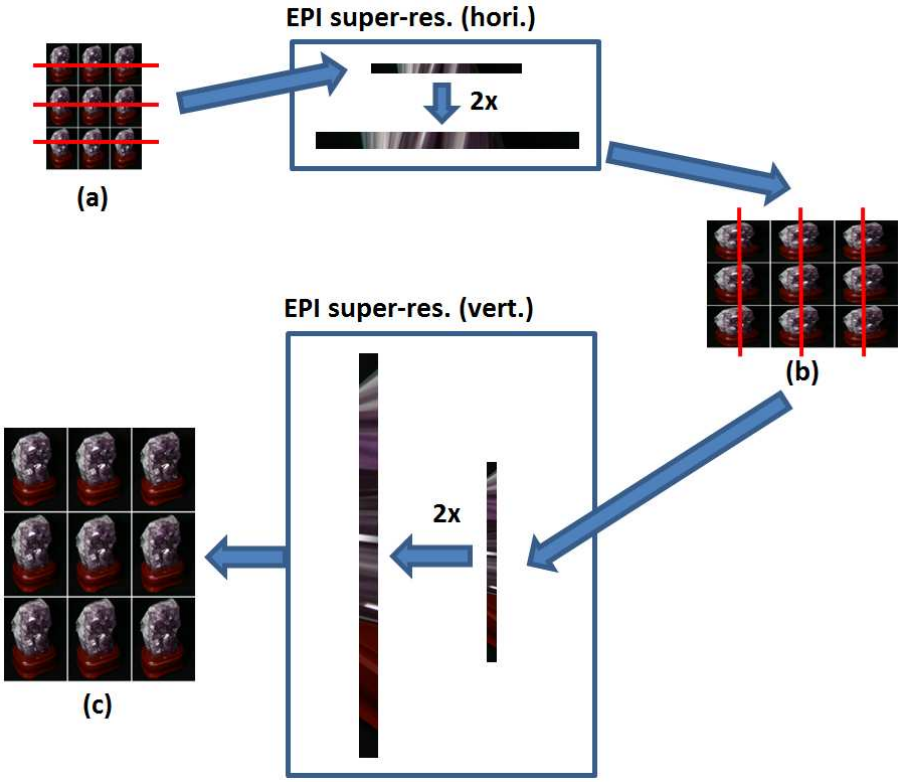


Figure 3.3: Joint angular/spatial super-resolution diagram

and will not be changed due to super-resolution, maintaining these constraints while well preserving the sharp edges in EPI-images ensures our image representations respecting the geometry. The previous works ignore this information.

3.4.2 Super-resolve EPI image

The key is to super-resolve each EPI-image. Fig 3.4 shows how different algorithms perform on 3× an EPI-image: Fig 3.4-(a) and (b) show the resulting horizontal EPI-image at $y = 398$ and $y = 451$, respectively; Fig 3.4-(c) show the resulting vertical EPI-images at $x = 806$. For each portion, we compare the following algorithm: bicubic interpolation, [79], [27] and [36].

From the figure, we can see that, 1) the bi-cubic interpolation tends to blur the EPI-image edge patterns, and the other three algorithms perform better in preserving those edges; 2) [79] and [27] seems performing oppositely: [79] blurs too much detail, and [27] exaggerates too much detail and [36] visually performs the best. We also compare the average of PSNR on our collected training EPI-images of the above methods: 37.6, 37.9, 39.5 and 39.8. Therefore, in this chapter, [36] will be used as our basic super-resolution tool for each EPI-image.

Also noticing that the broken line segment patterns in an EPI-image where occlusions

Algorithm 5 Joint angular/spatial super-resolution

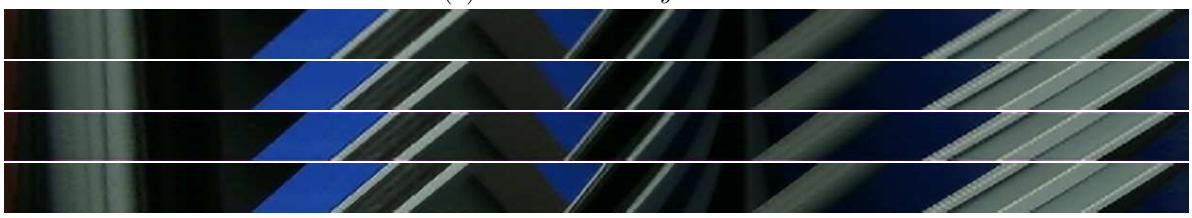
Require: All low-res light-field images $\{I_{(i,j)}\}$
Ensure: All super-resolved images $\{\hat{I}_{(i,j)}\}$
1: //**Hori**$_1$:
2: **for** each row (i) of the 2D camera array **do**
3: **for** each row (y) of the images **do**
4: Form horizontal EPI $E_h^{(i,y)}$ using $\{I_{i,1:M}\}$
5: Super resolve $E_h^{(i,y)}$ to get larger version $\tilde{E}_h^{(i,y)}$
6: Construct the horizontal reconstructed images for i-th row of 2D array $\hat{I}_{i,1:M}$.
7: **end for**
8: **end for**
9: //**Vert**$_1$:
10: **for** $j = 1 : M$ **do**
11: **for** each column x **do**
12: Form vertical EPI $E_v^{(j,x)}$ using $\{\tilde{I}_{1:N,j}\}$
13: Super resolve $E_v^{(j,x)}$ to get larger version $\tilde{E}_v^{(j,x)}$
14: **for** $i = 1 : N$ **do**
15: $\hat{I}_{(i,j)}(:, x) = \tilde{E}_v^{j,x}(:, i)$
16: **end for**
17: **end for**
18: **end for**

occur are preserved, see Fig 3.4. In other words, between the two viewpoints, some sort of occlusion event takes place. This information is particularly important when we want to synthesize novel views (virtual views). In order to ensure the generated virtual view has accurate information to make up the unobserved parts, we rely on good quality interpolation of the EPI-images. Therefore, our goal can be re-stated as: we want to super-resolve the EPI-images such that the global EPI-patterns are preserved in order to facilitate generating missing desired views. This is the best we can do – to synthesize sufficiently accurate information to faithfully reconstruct across occluded regions. With a smooth surface assumption, this relationship can still be maintained through magnified views; for the latter case, a new view will see something that camera 1 could see but camera 2 could not. If we assume the object surface is smooth enough, we can fill in the missing values for the middle camera between 1 and 2 by interpolation.

3.5 Experimental results

We test our algorithms, both joint image denoising and spatial/angular super-resolution, on the Stanford Light-field dataset [5]. The detailed information can be found in Chapter 2. To save computation, each image has been resized to one third of its original.

(a) 3X EPI - with $y = 398$



(b) 3X EPI - with $y = 451$

(c) 3X EPI - with $x = 806$

Figure 3.4: Comparison on single EPI-image super-resolution.

3.5.1 Joint image denoising

Our baseline algorithms are Elad et al. [27] [2] and [11] conducted on single images. Given the ground-truth images, we perturb all of them using Gaussian noise at varying noise levels: $\sigma = 25, 35, 50$, as our inputs.

Single image denoising performance

We first inspect the denoising performance on the central view image, $(9, 9)$-th camera. Table 3.1 shows the comparison of the peak signal-to-noise ratio (PSNR) with the baseline algorithms. Fig 3.6 presents the visual comparison of the denoised images from the different methods at injected noise level $\sigma = 50$. We can see that our algorithm works significantly better than [27] and [11] in almost all datasets, and only marginally poorer than any of the others in the remainder. With increasing strength at higher noise levels, our algorithm performs increasingly better. It can be seen that [27] and [11] exhibit decreasing performance at texture-less regions. Also comparing with the denoising performance of our method only operating using *cross*-structure of a light-field, the results here are very compatible.

[2] In our experiments, we follow their paper with 8×8 image patch sizes and dictionary size of 256 entries

To further check the influence of noise levels on the denoising performance, we compare a small window region on the denoised central view image using **Treasure** dataset, shown in Fig 3.5. We can see that: 1) both [27] and our method perform concretely better than [11]; 2) at lower noise level, [27] is compatable to ours, see Fig 3.5-(a); 3) at higher noise level, our method performs much better than [27], especially at the depth discontinuity and high-texture regions. The shaper edges are well preserved at those regions using our method, while are heavily blurred by [27]. This demonstrate the importance of the scene geometric information apart from the image appearance/color.

We also show the image denoising results for a high noise ($\sigma = 75$) situation and intermediate results, as a demonstration of the importance of each step, in Fig 3.7. As we expected, see Algorithm 4 and Fig 3.1, the horizontal ($Hori_1$, Fig 3.7-(f)) or vertical denoising stage ($Vert_2$, Fig 3.7-(h)) only can not fully remove the noise. However combining two directions, either $Hori_1 + Vert_1$ (Fig 3.7-(g)) or $Vert_2 + Hori_2$ (Fig 3.7-(i)) will be powerful enough to get satisfied results. Our final result, see Fig 3.7-(e), shows much better improvement over [27] and [11].

Joint image denoising performance

We also jointly denoise all 17×17 light-field images, with noise level $\sigma = 50$, some views of which are shown in Fig 3.8.

3.5.2 Spatial/Angular super-resolution

Using the same Stanford light-field dataset, we evaluate our angular/spatial super-resolution algorithm. Here, we compare the following algorithms: bi-cubic interpolation methods, Elad et al.[27], Kim et al.[36] and Yang et al.[79].

Single image super-resolution

We inspect the spatial super-resolution performance on a single image. We first compare the PSNR for one single image (middle $(9, 9)$-th) using different methods. Here, the image is super-resolved $3\times$ from its $\frac{1}{6}$ version. The results are shown in Table 3.2. In Fig 3.9, we visually compare our methods with the baseline algorithms. The more detailed comparison of different algorithms on a small region is shown in Fig 3.10. We can see that 1) all methods perform better than the simple bi-cubic interpolation in term of preserving the details; 2) the method in [79] produces a lot of ghosting blur around the strong edges; and 3) our algorithm performs better than [79], [27], and slightly better than [36] at the depth discontinuity and occlusion (see Fig 3.10).

Joint super-resolution results

To validate the joint super-resolution performance of our algorithm, we pick two small image patches from the low-res central image (Here, we are using the **Lego-Truck** dataset), shown

Image /	(σ)	**PSNR**		
		[27]	**[11]**	**ours**
Amethyst	25	30.8175	30.0549	**32.9654**
	35	29.1215	28.8256	**31.9475**
	50	27.393	27.391	**31.0316**
Beans	25	35.1374	33.4295	**36.0005**
	35	32.8649	32.5929	**34.8861**
	50	30.6304	29.0837	**33.2675**
Bracelet	25	**28.2739**	24.3033	27.1902
	35	**26.3248**	24.3087	26.0014
	50	24.2855	23.6278	**25.331**
Bunny	25	32.8815	30.8394	**34.0787**
	35	31.1023	30.3825	**33.1884**
	50	29.1187	27.8086	**32.0686**
Chess	25	31.2923	28.1903	**32.5317**
	35	29.1298	27.8923	**31.4029**
	50	26.9282	26.2984	**30.2529**
Flowers	25	32.37	30.4921	**33.4534**
	35	30.6279	30.2171	**32.5632**
	50	28.7291	27.9223	**31.7162**
Knights	25	30.0531	27.613	**30.4234**
	35	27.7266	25.7073	**29.6141**
	50	25.4665	25.2812	**28.3168**
Bulldozer	25	30.5095	29.6365	**30.6145**
	35	28.7356	28.0499	**29.6874**
	50	26.9881	26.6036	**29.0746**
Truck	25	31.8174	30.7166	**33.5237**
	35	29.9469	29.2646	**32.4453**
	50	27.8865	27.6979	**31.4091**
Treasure	25	**29.1872**	28.10	29.1071
	35	27.3456	26.5179	**27.9375**
	50	25.4713	25.064	**27.3453**

Table 3.1: Comparison of PSNR: denoising the central view

(a) Treasure ($\sigma = 25$)

(b) Treasure ($\sigma = 35$)

(c) Treasure ($\sigma = 50$)

Figure 3.5: Detailed denoising comparison for **Treasure** image at different noise levels. From left to right column: [27], [11], Ours

in Fig 3.11-top and Fig 3.12-top. After we perform the 3× super-resolution of all images, we display the corresponding **patch-cube**s on all 51×51 views produced. From the above figures, we observe smooth transition of the appearance of local patches after having super-resolved them. This demonstrates that we can successfully synthesizing the between-camera virtual views even at the places where depth discontinuities and occlusions occurred. Note the image patches we chose in low-resolution images have a lot of occlusion.

3.6 Conclusion and future work

In this chapter, we applied our image patch model in a light-field, as presented in Chapter 2, to two important light-field applications – joint image denoising and spatial/angular super-resolution. Given an image patch located in one camera, we leverage the full set of viewpoints of that patch, including the more accurate 3D geometric information and its ap-

Method	**bicubic**	[79]	[27]	[36]	**ours**
Ame	30.6953	31.7175	31.9341	**32.1469**	32.0932
Chess	30.5456	31.4806	31.9081	31.9624	**32.2246**
truck	31.2044	32.0392	32.2965	32.2046	**32.5524**
beans	38.6163	40.5772	**41.6229**	41.3542	40.1491
bracelet	25.5891	26.3443	26.5002	26.6314	**26.8347**
bull	29.1348	30.5684	31.0077	**31.2390**	30.9203
knights	28.9138	30.3265	30.8294	31.2192	**31.6194**
bunny	35.0439	36.9746	38.2272	**38.6603**	38.3359
flowers	31.1936	31.9906	32.0403	32.2708	**32.4328**
treasure	26.5258	27.0775	27.1975	27.2602	**27.8484**

Table 3.2: Comparison of PSNR: super-resolving the central view

pearance, to perform these two tasks. Our results will be not only more respectful of scene geometry, but more stable against image noise pollution as well. As the experimental results demonstrate, our joint image denoising algorithm performs significantly better than the baseline algorithms, with increasing strength as noise level rises. The angular/spatial super-resolution algorithm can more accurately span occlusions due to viewpoint change when synthesising novel perspectives. The experimental results demonstrate superior performance on both tasks.

The light-field camera provides a new type of data where the traditional single-view image can not. The complete description of an image patch, not only its appearance and color, but also scene geometry (implicitly encoded in the EPI-patches), provides a new perspective on many computer vision tasks, for example, depth recovery, image denoising, super-resolution, foreground segmentation, object detection and recognition, and so on. On this platform, all of the above algorithms can benefit from the highly redundant information in a light-field. Our future work aims to generalize our algorithms and apply them to some of these related computer visions tasks.

(a) Lego Truck

(b) Beans

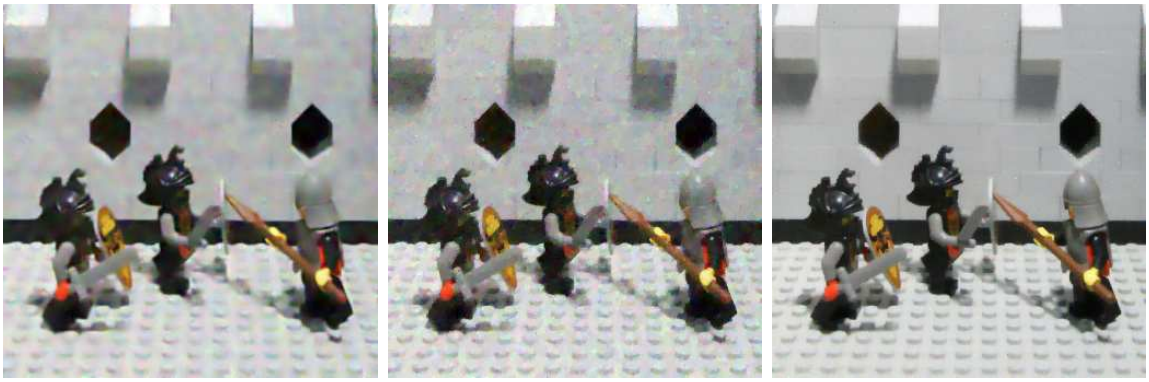
(c) Bracelet

(d) Bunny

Figure 3.6: Central view denoising comparison for image noise with $\sigma = 50$. From left to right column: [27], [11], Ours

(e) Chess

(f) Treasure



(g) Knights

(h) Lego Bulldozer

Figure 3.6: (continued)Central view denoising comparison for image noise with $\sigma = 50$. From left to right column: [27], [11], Ours

(i) Amethyst

(j) Flowers

Figure 3.6: (continued)Central view denoising comparison for image noise with $\sigma = 50$. From left to right column: [27], [11], Ours

(a) Original image

(b) Noisy image

(c) [27]

(d) [11]

(e) Our final

(f) $Hori_1$

(g) $Hori_1 + Vert_1$

(h) $Vert_2$

(i) $Vert_2 + Hori_2$

Figure 3.7: Results comparison at image noise level $\sigma = 75$. (a) and (b): reference image and noisy image; (c), (d) and (e): denoised result from [27], [11] and our final results. (f),(g),(h) and (i) show the intermediate results during our process – $Hori_1$, $Hori_1 + Vert_1$, $Vert_2$ and $Vert_2 + Hori_2$. PSNR: [11]=25.8170, [27]=25.4942, ours=29.2042.

Figure 3.8: Joint denoising results for several views at noise level $\sigma = 50$. From top to bottom and left to right: view of rows and columns 3, 9, and 15.

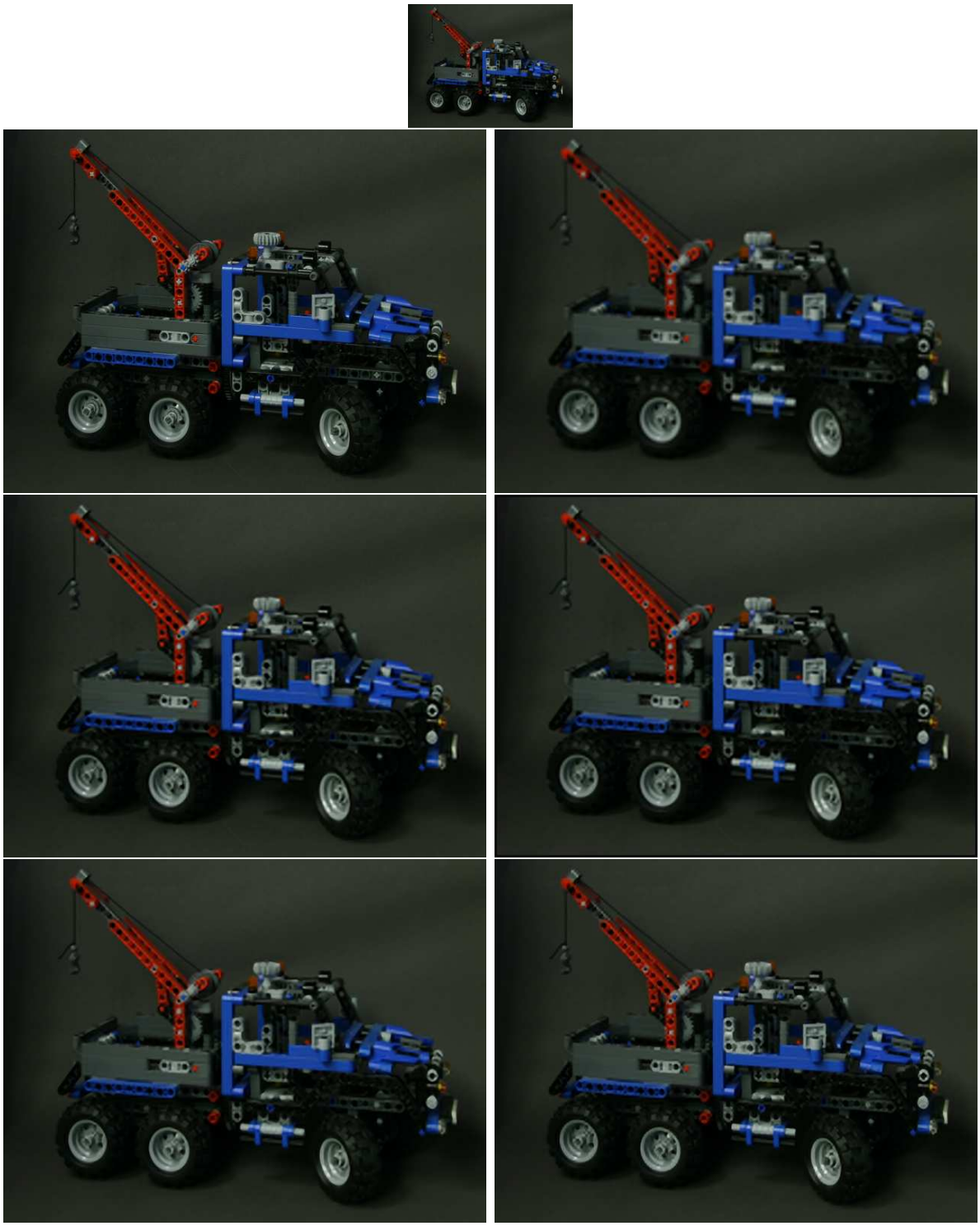
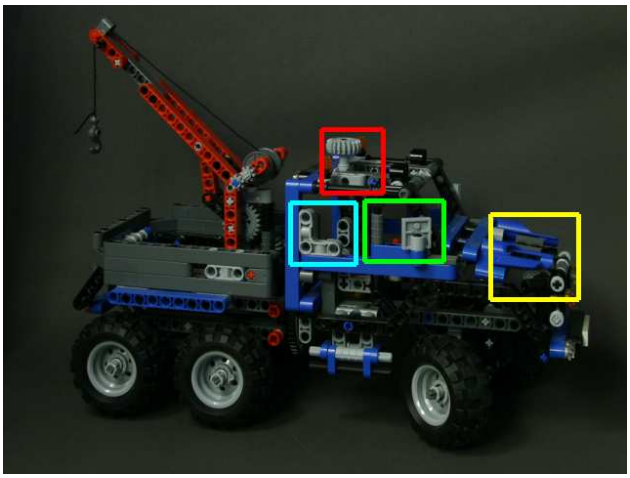
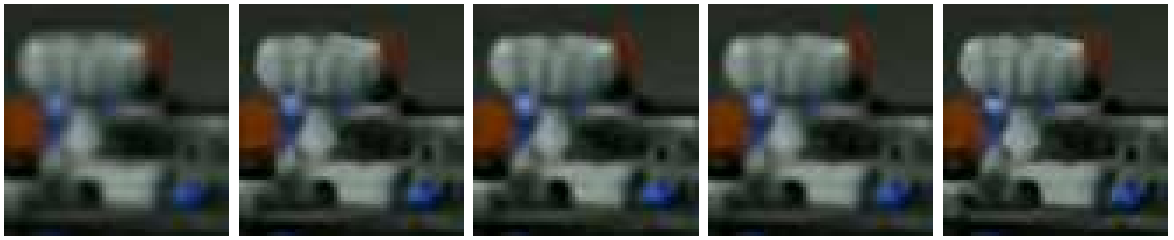


Figure 3.9: Single image super-res comparison: Low-res image, Ground truth, bicubic-interp., [79], [27], [36] and ours.

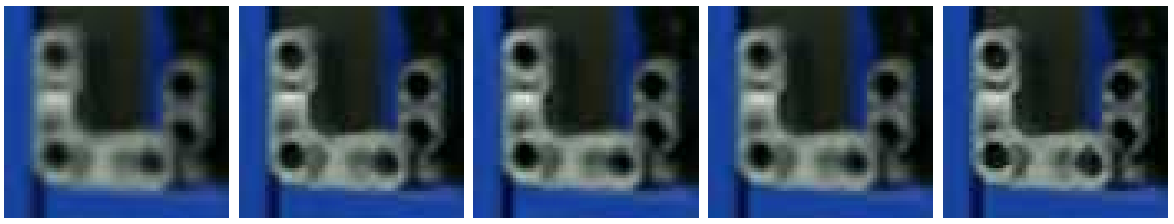


(a) Patches we are inspecting

(b) For patch labeled as "Green"



(c) For patch labeled as "Red"



(d) For patch labeled as "Cyan"

(e) For patch labeled as "Yellow"

Figure 3.10: Single image super-res comparison in detail. From left to right: bicubic interp., [36], [79], [27] and ours.

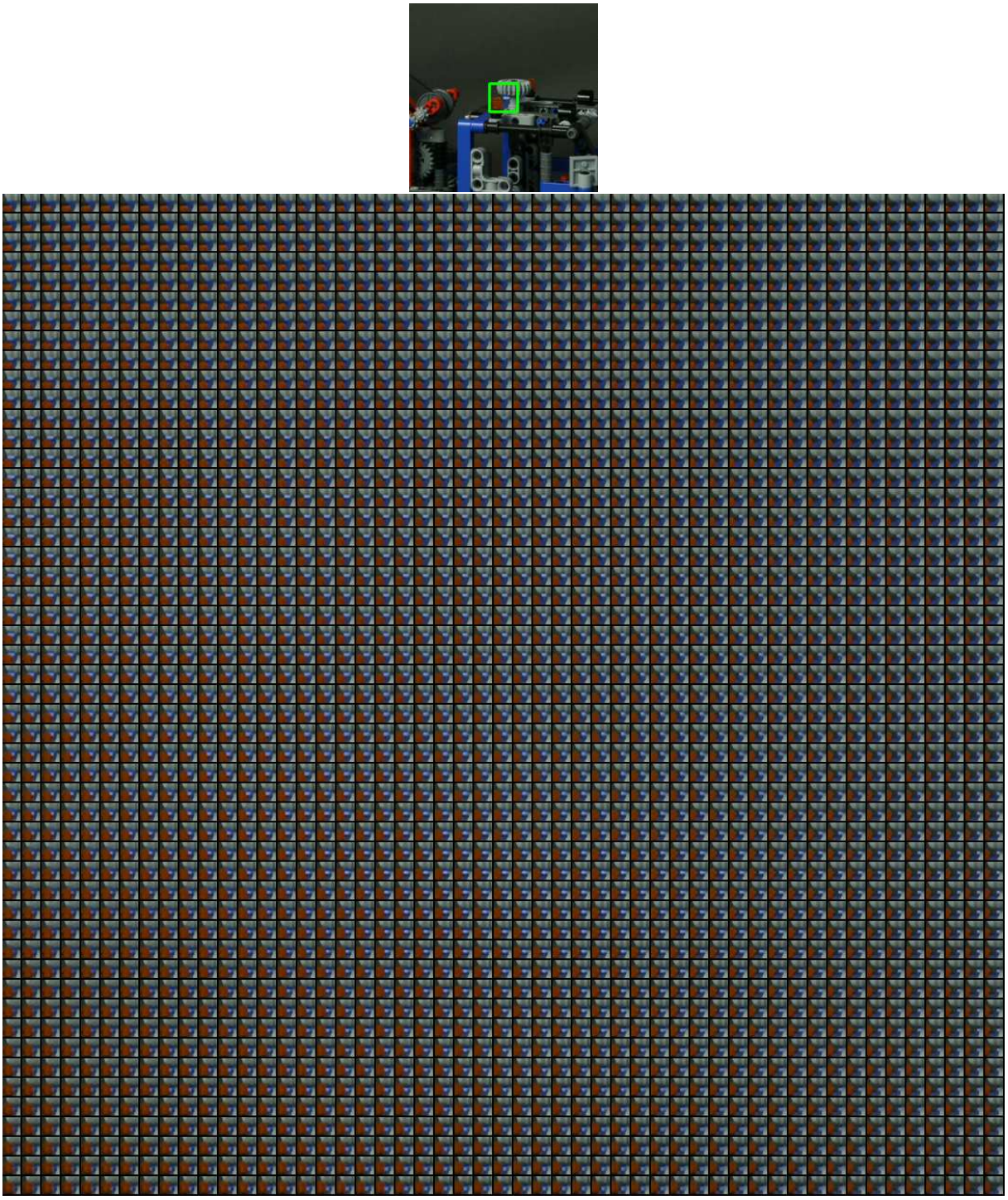


Figure 3.11: Reconstructed patches in the super-resolved light-field - 1

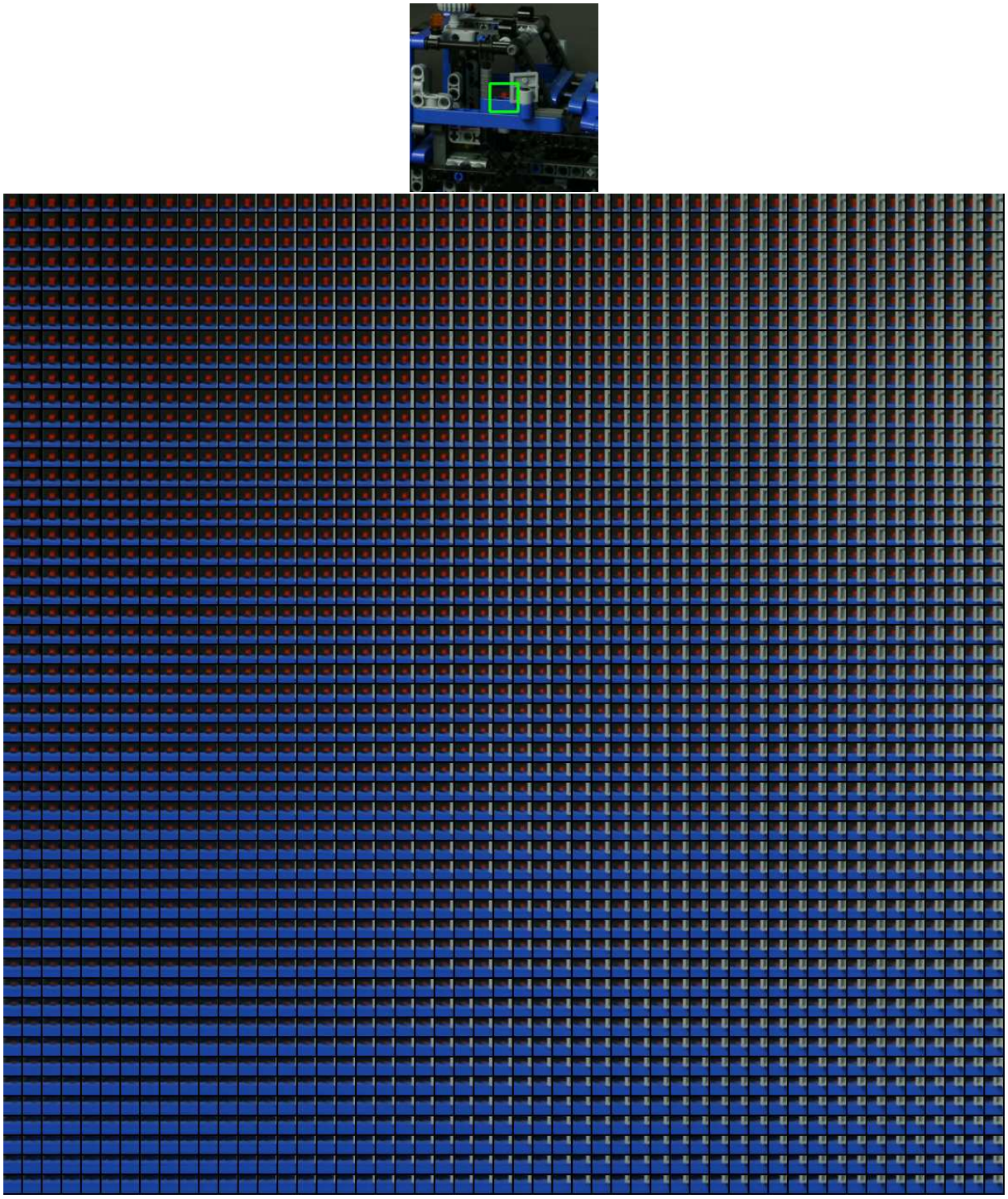


Figure 3.12: Reconstructed patches in the super-resolved light-field - 2

Chapter 4

Efficient KPCA feature extraction with Landmark Points

Inspired by the work of Coates et al. [2] and Zhang et al. [85], we view feature extraction from an image patch or EPI-patch in the context of the Kernel method [61] and the Nystrom sampling in this chapter. Given a point, we model its mapping on the original kernel eigen-space as a linear combination of some landmark points which span a reduced KPCA space. Depending on how the landmark points are chosen, the model can be estimated very efficiently, and we do not even need to compute and store the KPCA eigenvector approximations. Further, we show there exist close connections between this method and unsupervised feature learning framework recently proposed in [2][17].

4.1 Introduction

Kernel Principal Component Analysis (KPCA) [60] has been widely used in machine learning, for example nonlinear dimensionality reduction, manifold learning, image modeling [35] and etc, due to its capability of capturing the non-linear relationship in a high-dimensional space. Many problems can be also cast into the KPCA framework, for example locally linearly embedding [55], multi-dimensional scaling [23], spectral clustering [48]. However, the prohibitive computational cost, both in approximating the eigenvectors of the full kernel matrix and on out-of-sample extension of a new testing point, makes it impractical in extracting non-linear features for large-scale problems.

The literature deals with the first problem using low-rank matrix approximation or matrix completion, which assumes the full kernel matrix (K) is low-rank intrinsically. In the Nystrom sampling method proposed in Williams et al.[74], only a few columns of K are selected and used to perform eigen-decomposition, the eigen-pairs of the full kernel matrix can be recovered by an extra interpolation step. Talwalkar et al.[56] shows that even random column samples works pretty well in approximating the full kernel matrix. Other sampling strategies have also been proposed, such as kernel matching pursuit [53], low-rank matrix

approximation, greedy method, density-based sampling. The interested readers can refer to [56] and the references therein. [85] recently advocates to use the clustering centers from K-Means as landmark points which achieves better approximation performance.

The out-of-sample extension of KPCA is another source of huge computational overhead. Given a testing sample x, we need to store all training points and compute the pairwise distance of x to every training point, which is usually not affordable if the number of training samples is large. The common approach to solve this problem is to use a small number of representative landmark points which can approximate the KPCA eigenvectors well. [59], [65] tries to find these landmark points either from the existing training samples (reduced set selection) [54] or construct new vectors (reduced set construction) [20] in the original feature space. [4] and [6] formulate this problem as an optimization problem. However, none of the above methods gives satisfactory speedup.

We propose here an efficient KPCA feature extraction method using landmark points. The eigenvectors of the full kernel matrix are first approximated using cluster-based Nystrom sampling [85]. We then build a reduced KPCA space using landmark points such that the mapping of a point in the original KPCA space is preserved after we project it onto this reduced KPCA space. The combination of the two leads to a very efficient KPCA feature extraction for large scale problems. If combined with convolution-type structure in patch-based image modeling, our model can learn more robust features and achieve state-of-the-art recognition performance in two face recognition benchmark datasets. We also compare different strategies on how to choose landmark points. Our method is also similar to Kernel Dependence Estimation [73]

Different from other methods, such as sparse coding [52], which defines a good feature as a sparse vector which can reconstruct the original signal well, we argue that good features are those which can preserve the pairwise distances, which in turn can approximate the kernel matrix well. In this chapter, we focus on the connection between KPCA and the popular unsupervised feature learning algorithms, such as the Bag-of-Words (BoW) model [63] and the unsupervised feature learning framework proposed in Coates et al. [2] and Boureau et al.[17]. We show that the cluster centers or sparse coding dictionary atoms used in those papers are actually some representative landmark points. From the perspective of Nystrom sampling, we observe: 1) even random samples from the training set can be used as landmark points, and 2) the more landmark points we choose, the better the feature quality, which is consistent with the observations made in [2][17]. Realizing the important roles of choosing landmark points in unsupervised feature learning, we will further investigate alternative ways to choosing the landmark points, which will be the main topics of Chapter 5.

The chapter is organized as follows: we first briefly introduce the Kernel PCA and Nystrom sampling methods [74] in Section 4.2. Then our out-of-sample projection based on landmark points for KPCA is presented in Section 4.3. The experimental results and conclusion are presented in Section 4.4 and 4.5, respectively.

4.2 Kernel Principle Component Analysis and Nystrom sampling

Given a sample of N points, $X = \{x_1, \ldots, x_N\}$, where $x_i \in \mathbb{R}^d$, KPCA first maps the data into a high-dimensional (even infinite) feature space using a nonlinear mapping $\phi(x)$, and then performs PCA in that space. Suppose the mapped data matrix is

$$\Phi_X = [\phi(x_1), \ldots, \phi(x_N)]$$

and the eigen-decomposition of the covariance matrix $\Sigma_\phi = \Phi_X \Phi_X^T$ is $\Sigma_\phi = USU^T$, then the projection of a new point can be written as:

$$y = U^T \phi(x)$$

Since it is impossible to calculate Σ_ϕ if we don't know ϕ, we turn to finding the gram matrix or kernel matrix $K = \Phi_X^T \Phi_X$, where $K(i,j) = \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$ and $k(\cdot, \cdot)$ is a kernel function. If the kernel matrix can be decomposed as:

$$K = V \Lambda V^T$$

then

$$U = \Phi_X V \Lambda^{-\frac{1}{2}} \tag{4.1}$$

therefore the projected point for x_0 can be represented as:

$$y = \Lambda^{-\frac{1}{2}} V^T k(x_0, \cdot)$$

where $k(x, \cdot) = [k(x, x_1), \ldots, k(x, x_N)] \in \mathbb{R}^{N \times 1}$.

In the above, we assume the data in the feature space has been centered, which can be achieved by:

$$\tilde{K} = HKH$$

where H is the centering matrix

$$H = \mathbb{I} - \frac{1}{N} \mathbf{1}\mathbf{1}^T$$

$\mathbb{I}$ is an identity matrix and $\mathbf{1}$ is all-one column vector. Similarly, a testing point $\phi(x)$ can be centered as following:

$$\tilde{k}(x_0, \cdot) = k(x_0, \cdot) - \mathbf{1}\frac{1}{N} k(x_0, \cdot)^T \mathbf{1} - \frac{1}{N} K\mathbf{1} + \frac{1}{N^2} \mathbf{1}^T K \mathbf{1}\mathbf{1} \tag{4.2}$$

where $K \in \mathbb{R}^{N \times N}$ is the full kernel matrix.

Depending on the applications, different kernel functions $k(\cdot, \cdot)$ can be used [61], such as Gaussian kernel, polynomial kernel, linear kernel, etc.

4.2.1 Low-rank matrix approximation of the Kernel matrix

The central task of KPCA is to calculate the eigen-pairs of the kernel matrix K. For a large-scale problem, it is very hard to directly perform eigen-decomposition on the $N \times N$ matrix. One way is to generate its low-rank approximation $\hat{K}$ of K based on some criteria:

$$K \approx \hat{K} = LL^T \tag{4.3}$$

where the rank of L is $m \ll N$. The quality of the approximation is measured by the reconstruction error on K: $\text{err} = ||K - \hat{K}||_F$. Among many algorithms, the Nystrom sampling [74] method uniformly samples $m \ll N$ columns of K. Without loss of generality, the K matrix can be rearranged based on the sampling as follows:

$$K = \begin{pmatrix} K_{mm} & K_{m,N-m}^T \\ K_{m,N-m} & K_{N-m,N-m} \end{pmatrix}$$

where

$$L = \begin{pmatrix} K_{mm} & K_{m,N-m}^T \end{pmatrix}^T$$

Then the original kernel matrix K can be approximated by:

$$K \approx EW^{-1}E^T \tag{4.4}$$

and the eigen-pair of K can be approximated by:

$$V_{ny} = \Lambda_w^{-1} E^T U_w, \quad \Lambda_{ny} = \Lambda_w \tag{4.5}$$

where (U_w, Λ_w) are the eigen-pair of the matrix $W \in \mathbb{R}^{m \times m}$, up to scale. E is called an extrapolation matrix, which extends the eigenvector of W to the whole kernel matrix. Since we only need to perform the eigen-decomposition on $m \times m$ matrix W, the computational overhead is reduced dramatically. Also if the rank of K is equal to m, then the approximation is accurate [74]. Different sampling strategies have been tried.

4.3 Proposed method

In this section, our method is proposed for efficient KPCA feature extraction, which proceeds by applying cluster-based Nystrom approximation [85] followed by a modified reduced-set method, which results in a very efficient method to extract non-linear features of KPCA. Also we will explore the connection of our method with the current unsupervised feature learning framework[2][17], including the Bag-of-Word model [63].

4.3.1 Eigenvector approximation

Instead of randomly choosing some columns of the kernel matrix to construct the matrix E and W in Eqn 4.4, Zhang et al. [84][85] presented a novel method to approximate the

eigen-pairs of K using K-Means clustering. Some theoretical analysis shows the reconstruction error of the full kernel matrix is bounded by the quantization errors of the K-Means clustering.

Given the m K-Means cluster centers, $\{c_1, \ldots, c_m\}$, we approximate the full kernel matrix as follows:

$$K \sim LL^T, \quad \text{where } L = EW^{-\frac{1}{2}}$$

where $E \in \mathbb{R}^{N \times m}$ and $E_{ij} = k(x_i, c_j)$, $W \in \mathbb{R}^{m \times m}$ and $W(i,j) = k(c_i, c_j)$. The eigen-pairs of K can be estimated as:

$$V_{km} = LV_G\Lambda_G^{-\frac{1}{2}}, \quad \Lambda_{km} = \Lambda_G \tag{4.6}$$

where (V_G, Λ_G) is the eigen-pair of $G = L^TL$:

$$G = V_G\Lambda_G V_G^T$$

In order to obtain the eigenvectors of the centered kernel matrix, $\tilde{K} \approx HKH$, we have the following:

$$\tilde{V}_{km} = \tilde{L}\tilde{V}_G\tilde{\Lambda}_G^{-\frac{1}{2}}, \quad \tilde{\Lambda}_{km} = \tilde{\Lambda}_G \tag{4.7}$$

where $(\tilde{V}_G, \tilde{\Lambda}_G)$ are the eigen-pair of the matrix $\tilde{G} = (HL)^T(HL) = L^THL$ and $\tilde{L} = HEW^{-\frac{1}{2}}$. Using Eqn 4.1, the eigenvectors of the centered kernel matrix can be written as:

$$\tilde{U}_{km} = \tilde{\Phi}_X\tilde{V}_{km}\tilde{\Lambda}_{km}^{-\frac{1}{2}} \tag{4.8}$$

Noticing $\tilde{V}_{km} \in \mathbb{R}^{N \times m}$, actually we can get its smaller version by throwing away the eigenvectors corresponding to very small eigenvalues. Also it will be clear from the next section that we actually do not need to calculate and store $\tilde{V}_{km}$.

The above method provides an efficient way to approximate the eigenvectors of the full kernel matrix. The key observation is that the Nystrom low-rank approximation depends crucially on the quantization error induced by encoding the training dataset with the cluster centers. This method is suitable for large-scale problems. Another advantage is that the eigenvectors found using this method are orthognomal, see Eqn 4.7, which has been shown to be better than the non-orthognomal counterpart, for example Eqn 4.5.

4.3.2 Out-of-sample projection using landmark points

Although the eigenvector of the full kernel matrix can be approximated very efficiently using the above method [84], to perform out-of-sample projection for a testing point x_0, we still need to keep all training samples and calculate the pairwise distance between x_0 and all the training samples, which is impractical for large-scale problems. In this section, we first construct a reduced feature space spanned by a few landmark points which could have different center for speedup reason. A linear projection P is used to associate the two feature spaces such that we have very similar kernel mapping on both feature spaces for a given testing point.

For a sample x_0, its projection onto the approximated eigen-space can be calculated:

$$z(x_0) = \tilde{U}_{km}^T(\phi(x_o) - M) = \tilde{\Lambda}_{km}^{-\frac{1}{2}}\tilde{V}_{km}^T[\tilde{k}^M(x_0, \cdot)] \tag{4.9}$$

where $z(x_0) \in \mathbb{R}^{m\times 1}$, $M = \frac{1}{N}\sum_i \phi(x_i)$ is mean of all samples in the original feature space, $\tilde{k}^M(x_0, \cdot) \in \mathbb{R}^{N\times 1}$ is the centered version of $k(x, \cdot) = [k(x_0, x_1), \ldots, k(x_0, x_N)]^T$ with respect to the sample mean M. In this chapter, a simple linear model is proposed to approximate the kernel mapping of $z(x_0)$ using the reduced feature space spanned by $n \ll N$ landmark points $(l_0, \ldots, l_n)$:

$$\hat{z}(x_0) = P^T[\tilde{k}_l(x_0, \cdot)] + b \tag{4.10}$$

where $\hat{z}(x_0) \in \mathbb{R}^{m\times 1}$, $P \in \mathbb{R}^{n\times m}$. $k_l(x_0, \cdot) = [k(x_0, l_1), \ldots, k(x_0, l_n)]^T \in \mathbb{R}^{n\times 1}$, and $\tilde{k}_l(x_0, \ldots)$ is some normalized version of $k_l(x_0, \ldots)$, which will be clear later in this section. And also the bias term b is omitted here.

To solve for P, we try to minimize the Euclidean distance between the actual projection $z(x_i)$ and its approximated version $\hat{z}(x_i)$ for all the training samples, and form the following optimization problem:

$$P^* = \text{argmin}_P \quad J = \left|\left|\tilde{\Lambda}_{km}^{-\frac{1}{2}}\tilde{V}_{km}^T\tilde{K}^M - P^T\tilde{K}_l\right|\right|_F^2 + \lambda||P||_F^2 \tag{4.11}$$

where $\tilde{K}_l = \left[\tilde{k}_l(x_1, \cdot), \ldots, \tilde{k}_l(x_N, \cdot)\right] \in \mathbb{R}^{n\times N}$, $\tilde{K}^M \in \mathbb{R}^{N\times N}$ is the centered original kernel matrix. The second regularization term is used to prevent numerical instabilities. Solving this optimization function, we can get:

$$P^* = \left[\tilde{K}_l \cdot (\tilde{K}_l)^T + \lambda\mathbb{I}\right]^{-1}\left[\underbrace{\tilde{K}_l\tilde{K}^M \cdot \tilde{V}_{km}}_{*}\right]\tilde{\Lambda}_{km}^{-\frac{1}{2}} \tag{4.12}$$

where $\mathbb{I}$ is an identity matrix.

The computation cost of using Eqn 4.12 comes from the part marked as "*". Noticing $\tilde{K}^M \in \mathbb{R}^{N\times N}$, $\tilde{K}_l \in \mathbb{R}^{m\times N}$ and $\tilde{V}_{km} \in \mathbb{R}^{N\times m}$, we need to keep a copy of $N \times N$ matrix and also the complexity is $O(N^3)$. Actually depending how we construct our reduce feature space, including choosing the landmark points and the centering matrix, Eqn 4.12 can be significantly simplified accordingly:

- Using the same cluster centers $\{c_1, \ldots, c_m\}$ as our out-of-sample projection landmark points, and further we center all points with respect to M using Eqn 4.2, Eqn 4.12 can be simplified as:

$$P^* = \tilde{V}_G \tag{4.13}$$

Above, we use the facts: $\tilde{K}_l = \tilde{L}^T$, $\tilde{K}^M \approx \tilde{L}\tilde{L}^T$ and $\tilde{L}^T\tilde{L} = \tilde{V}_G\tilde{\Lambda}_G\tilde{V}_G^T$. It is easy to prove that the reconstruction error in Eqn 4.11 is zero in this case. That is, we can perfectly reconstruct our kernel mapping in our reduced feature space. Given a testing point x_0, however, we have to calculate the full kernel matrix K when applying Eqn 4.2 to center its image $\phi(x_0)$, which is impractical for large-scale problems;

- Using the same K-Means center, or uniformly sampled n training samples, as our landmark point for out-of-sample projection. But for speed reasons, we center all samples using the sample mean of the landmark points only: $m = \frac{1}{n}\sum_i \phi(l_i)$. More specifically, for a given point x_0, we get its centered version by:

$$\tilde{k}^{m_l}(x_0, \{l_k\}) = k(x_0, \{l_k\}) - \mathbf{1}\frac{1}{m}\mathbf{1}^T k(x_0, \{l_k\}) - \frac{1}{m}K_{landmark}\mathbf{1}^T + \frac{1}{m^2}\mathbf{1}^T K_{landmark}\mathbf{1}\mathbf{1}^T \tag{4.14}$$

  where $K_{landmark} \in \mathbb{R}^{n\times n}$, and $K_{landmark}(i,j) = k(l_i, l_j)$. Comparing with Eqn 4.2, it is very easy to compute and store $K_{landmark}$ when $n \ll N$, which will dramatically speed up the out-of-sample projection. Then Eqn 4.12 can be simplified as:

$$P^* = \left[\tilde{K}_l \cdot (\tilde{K}_l)^T + \lambda\mathbb{I}\right]^{-1} \tilde{K}_l \tilde{L} \tilde{V}_G \tag{4.15}$$

  Here, we again use the fact: $\tilde{K}^M \approx \tilde{L}\tilde{L}^T$, and $\tilde{L}^T\tilde{L} = \tilde{V}_G\tilde{\Lambda}_G\tilde{V}_G^T$.

We will experimentally examine and compare these strategies in our experimental section 4.4.

4.3.3 Analysis

Our out-of-sample model bears some similarities with the reduced set method [59] [65], which is a common method to get a sparse representation in order to speed up the kernel method. Eqn 4.9 is equivalent to:

$$\tilde{U}_{km} \approx \left[\begin{array}{cccc} \tilde{\phi}(l_1) & \tilde{\phi}(l_2) & \ldots & \tilde{\phi}(l_n) \end{array}\right] P$$

That is, each eigenvector of the kernel matrix $\tilde{K}$ is assumed to be a linear combination of the few landmark points. The landmark points are either chosen from the original training set or constructed by pre-image optimization [59]. Different from [59], we construct our reduced feature space not only using landmark points but also using different centering matrix, which leads to very efficient out-of-sample projection suitable for large-scale problems.

The efficiency of our method roots in merging kernel matrix approximation using cluster-based Nystrom sampling [85] and the modified reduce-set method. More specifically, if we use the Nystrom sampling method [74] to approximate the kernel eigenvectors, plugging Eqn 4.5 into Eqn 4.12 and using the fact $K \approx EW^{-\frac{1}{2}}E^T$, we have:

$$P^*_{ny} = \left[\tilde{K}_l \cdot (\tilde{K}_l)^T + \lambda\mathbb{I}\right]^{-1} \tilde{K}_l \left[\tilde{E}W^{-\frac{1}{2}}\tilde{E}^T \cdot \tilde{V}_{ny}\tilde{\Lambda}_{ny}^{-\frac{1}{2}}\right] \tag{4.16}$$

We can see that computational overhead is twice that of our method (see Eqn 4.15). Another benefit is that we do not even need to compute and store the kernel eigenvector (Eqn 4.7), which is obvious in Eqn 4.15.

Our method can be regarded as a general version of the Bag-of-Words model based on soft-Kmeans clustering. In the BoW model, K-means clustering is first applied on image features, such as SIFT/HoG, to get the cluster centers $\{c_1, \ldots, c_m\}$, each new feature x_o can then be coded using those centers:

$$\text{code}(x_0) = \left[\exp(-\frac{||x_0 - c_1||}{\sigma^2}) \quad \exp(-\frac{||x_0 - c_2||}{\sigma^2}) \quad \exp(-\frac{||x_0 - c_m||}{\sigma^2}) \right]$$

This soft-Kmeans coding strategy has gained success in many computer vision tasks. Although it has some probabilistic interpretation (such as Gaussian Mixture Model), it still lacks some justification of why it works. In this section, instead of directly using the above equation, we argue that we should perform two extra steps: centering and projection, as suggested by Eqn 4.13. Even Eqn 4.13, as shown here, is a very rough approximation of the KPCA projection by assuming the full kernel matrix with rank m. An even better approximation can be acquired by Eqn 4.12.

Putting the Bag-of-Words model in the setting of the Nystrom sampling framework, we show that cluster centers are actually landmark points. The more points we choose, the better we can approximate the full kernel matrix, and the better is the feature quality. Also even some random samples from the training set can be selected as the landmark points in order to get reasonable features. These observations are confirmed by [2][17].

4.4 Experimental results

We tested our algorithm on several publicly available datasets: Ionosphere, Dorothea, Gisette from the UCI machine learning repository [1], and two face recognition datasets: Yale extended B [30] and AR face database [46].

For the two face recognition experiments, we adopt the convolution-type patch-based representation for each image [2]. The projection matrix P is first learned from randomly sampled $30,000$ patches from all the training images. Given a testing image, we calculate the KPCA feature z for each sliding window patch and then form a sparse vector by applying:

$$f(z) = \max(0, z - \bar{z})$$

where $\bar{z}$ is the mean of z. Those sparse vectors are spatially averaged over a 2×2 grid on that image, and the final vectors is constructed by concatenating the four vectors. The average pooling strategy can handle slight image shift, deformation, and also is robust to noise. Our baseline algorithm is [2], where each patch is coded using its Euclidean distances to all cluster centers.

4.4.1 UCI dataset

The three UCI datasets used in this chapter are summarized in Table 4.1. All of the three dataset are binary classification problems. **Ionosphere** is to identify whether some type

Dataset	# train	# test	# features
Ionosphere	175	176	34
Dorothea	800	350	100000
Gisette	6000	1000	5000

Table 4.1: UCI dataset used in this chapter. For Ionosphere, we randomly split 50% as training and testing set. For Dorothea and Gisette, we report the classification accuracy on the validation set.

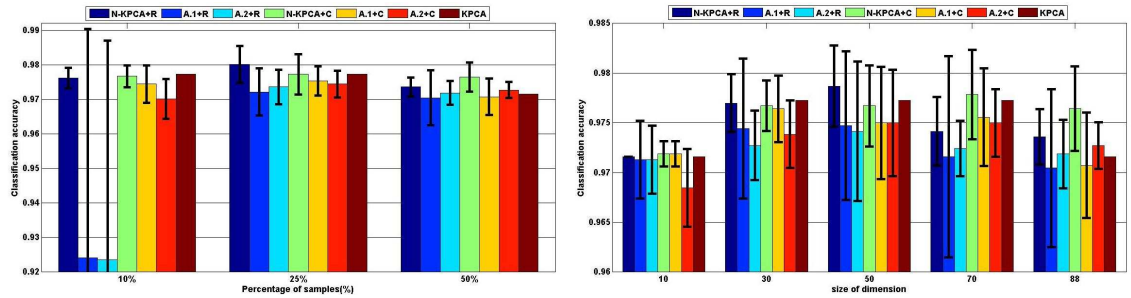
of structure exists in the ionosphere. **Dorothea** is a drug discovery dataset, the goal is to classify a Chemical compound represented by structural molecular features as active or inactive. The **Gisette** dataset challenge is to separate the digits '4' and '9'.

The following feature extraction methods are compared, "R"/"C" represents using random samples or cluster centers as landmark points:

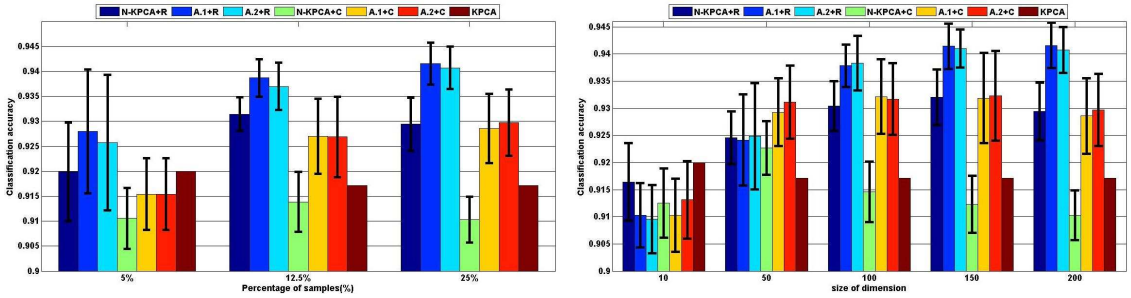
- N-KPCA + R/C: the kernel eigenvectors are estimated by Nystrom sampling KPCA [74], and the out-of-sample projection uses all training points;
- A.1 + R / C: the kernel eigenvectors are approximated by Nystrom sampling KPCA [74], the same landmark points are used for out-of-sample projection, see Eqn 4.16;
- A.2 + R / C: the kernel eigenvectors are approximated by cluster-based Nystrom sampling [85], the same landmark points are used for out-of-sample projection, see Eqn 4.15;
- KPCA: we use full KPCA model to extract features;

After extracting features for each sample, we apply linear SVM for classification. The results on UCI datasets are summarized in Fig 4.1. From the above results, we have the following observations:

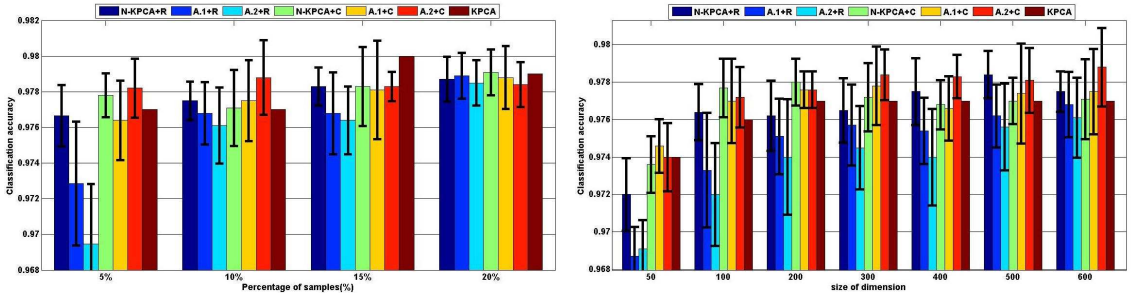
- Our approximation methods achieves classification accuracy similar to Nystrom-KPCA and KPCA; On some datasets, e.g., **Giestte**, our method is even better than KPCA – one possible reason is the denoising effects; and our method is cheaper to compute;
- The classification performance of randomly-chosen landmark points or cluster centers is very close. On the **Dorothea** dataset, A.1 is systematically better than A.2. How to choose reasonable landmark points depends on the characteristic of the dataset we use;
- The model with "cluster centers" performs better when the number of samples is very small, for example, 5%. In real applications, this is usually what we expect.



(a) Ionosphere dataset



(b) Dorothea dataset



(c) Gisette dataset

Figure 4.1: Classification accuracy on Ionosphere,Dorothea and Gisette dataset. Left: classification accuracy with number of samples ratio (m/N); Right: classification accuracy with number of PCA dimension (d).

- Depending on the dataset, we can choose fewer kernel eigenvectors without losing much accuracy, which further reduces the computational cost. E.g., when we choose $d = 400$ for the **Gisette** dataset, we can reach similar classification as with $d = 600$.

# sampl.	[2]	A.2+C	A.2 + R
30	**94.1987±1.0734**	93.0926±0.9350	91.0128±1.0409
84	97.0367±0.4938	**97.9062±0.6312**	97.8854±0.5798
144	97.8297±0.3098	98.7396±0.4022	**98.8147±0.3017**
300	98.2958±0.5496	**99.2696±0.1338**	99.2487±0.1631
500	98.7897±0.3010	**99.5270±0.1524**	99.4783±0.1597

Table 4.2: Classification accuracy for YaleB database

4.4.2 Yale Extended Face database B

The Extended Yale B database is a face recognition benchmark dataset, it contains 2414 frontal-face images of 38 individuals. The cropped and normalized 192×168 face images were captured under various laboratory-controlled lighting conditions. Each image is down-sampled to the size of 48×42, without any other preprocessing. The patch size we used is 12×12.

We follow the similar experimental protocol of [75][82]: For each subject, we randomly select half of the images for training (32 images per subject), and the other half for testing. We run the above processes 10 times, and report the average classification rate and its standard deviation in Table 4.2. In our experiments, we compare two landmark point-selection strategies ("R": random samples, or "C": K-Means clustering center), a linear SVM is used as our classifier.

From Table 4.2, we have the following observations:

- The more landmark points we choose, the higher is the classification rate. That is, the quality of the extracted feature is closely correlated with how good we can approximate our kernel matrix, and KPCA can effectively extract non-linear features from the high dimensional data;
- The performance of "A.2+C" is systematically better than "A.2 + R", which means that carefully choosing landmark points can definitely improve classification performance;
- If we choose the number of samples to be 600, our **A.2** can reach 99.5% and KPCA+ny is 99.7% accuracy;
- The state of the art accuracy reported is 99.4% [82]. They use a carefully designed weighted sparse coding model to extract features, and all training samples are treated as dictionary atoms, which is very expensive to compute.

#samp.	[2]	A.2 + C	A.2 + R
30	**92.56±0.6**	92.19±1.2	91.67±1.1
54	94.99±0.8	**95.59±0.7**	94.83±0.8
150	96.30±0.5	97.59±0.4	**97.7±0.4**
500	96.90±0.4	**98.63±0.3**	98.47±0.4
800	97.2±0.4	**99.1±0.2**	99.0±0.2

Table 4.3: Classification accuracy for AR database

4.4.3 AR face database

The AR database consist of over 4000 frontal images for 126 individuals. For each individual, 26 pictures were taken in two separate sections. These images include more facial variations including illumination change, expressions, facial disguises, and fewer training samples per subject compared to the Extended Yale B database. In the experiment (similar to [75][82]), we choose a subset of the dataset consisting of 50 male and 50 females subjects. For each subject, 14 images with only illumination change and expressions were selected: the seven images from section 1 for training, and the other seven from section 2 for testing. The images are downsampled with dimension 165×120 and converted to grayscale.

Following the similar experimental protocols of [75], [82], three experiments are conducted to test our KPCA feature extraction. The patch size used is 15×15.

Only considering illumination and expression

In this experiment, we only consider the faces with illumination and expression changes. We run our experiments 30 times, and report the average classification rate and its standard deviation, shown in Table 4.3. Again, we observe that our method is systematically better than [2] in most cases. The carefully chosen landmark points helps classification performance. As reference, the-state-of-the art accuracy is 96.0% in [82], which uses a robust sparse coding model.

Face recognition with real disguise

We use the same experimental setting as in [75][82]. A subset AR database is used in this experiment.

- disguise test 1: we train our model on the non-occluded frontal views (with different facial expressions), and test it on the face images with sunglasses and scarf. The results are summarized in Table 4.4;
- disguise test 2: We train our model on non-occluded frontal faces (with different illuminations) in Section 1, and test our model on the disguised images (with various

Alg.	SRC [75]	GSRC [81]	RSC [82]	[2]	Ours
Glass	87.0%	93.0%	99.0%	80.3±0.8	**99.7%±0.26**
Scarf	59.5%	79.0%	**97.0%**	86.5±0.7	96.7 ± 0.67

Table 4.4: Classification accuracy for AR face database with disguise test 1

Alg.	glass-sec1	scalf-sec1	glass-sec-2	scalf-sec2
SRC[75]	89.3%	32.3%	57.3 %	12.7
GSRC[81]	87.3%	85%	45%	66%
RSC[82]	94.7%	91.0 %	80.3%	72.7 %
[2]	73.7083±5.53	77.0±0.9428	46.0 ±4.2426	44.3333±0.9428
Ours	**98.9333±0.3651**	**93.4167±1.0351**	**83.7917±0.5893**	**78.7333±1.8012**

Table 4.5: Classification accuracy for AR face database with disguise test 2.

illuminations and sunglasses or scarf) for both sections. Table 4.5 shows the comparison. In this experiment, patch size is 15×15, the number of cluster centers used are 1000, 1000, 1500 and 1500, respectively).

From the above two tables, we can see our method performs better than the state of art algorithms in these two experiments. One of the reasons is that our KPCA feature extraction method, as a better BoW model, can capture more robust and meaningful image features. On the other hand, the patch-based image representation we use is suitable for these classification tasks. Although about 40% of the face is blocked, the average-pooling strategy can still pick up enough information to distinguish faces.

4.5 Conclusion and future work

We have presented a very efficient method for Kernel PCA feature extraction based on landmark points for large scale problems. The experimental results demonstrate the effectiveness and efficiency of our method. Also the current success of the unsupervised feature learning framework can be explained by our method. We show that carefully selected landmark points are important to getting robust and more meaningful image features when the number of landmark points is moderate. The following chapter will investigate alternative ways to select better landmark points in order to further improve feature quality.

Chapter 5

GEV-clustering

Recent studies have shown that K-means, with larger K, can effectively learn local image patch or EPI-patch features; accompanied with appropriate pooling strategies, it performs very well in many visual object recognition tasks. An improved K-means cluster algorithm, GEV-Kmeans, based on the Generalized Extreme Value (GEV) distribution, is proposed in this chapter. Our key observation is that the squared distance of a point to its closest center adheres to the Generalized Extreme Value (GEV) distribution when the number of clusters is large. Differing from the K-means algorithm, we minimize the reconstruction errors by ignoring those points with lower GEV probabilities (i.e. rare events), and focus on others points which might be more critical in characterizing the underlying data distribution. Consequently, our algorithm can handle outliers very well in situations where the conventional K-means algorithm suffers. Experimental results demonstrate the effectiveness of our algorithm.

5.1 Introduction

In order to perform a high-level visual task, an unsupervised feature learning framework has been proposed [2][34], attaining much success in many visual object recognition tasks. A feature learning algorithm is employed over a large number of randomly sampled local image patches in order to learn feature representations. The learned features, after some non-linearity and spatial pooling (average or max-pooling), are fed into a linear SVM for classification. [2] has shown that, after appropriate pre-processing, K-means clustering performs surprisingly well in comparison with other more sophisticated models, such as Restricted Boltzmann Machine [33], sparse coding [51], etc. [2] also shows that classification performance increases as the number of cluster centers used in the K-means increases.

Given a set of points in a high-dimensional space, the goal of the K-means algorithm (Lloyd's algorithm) [42] is to find cluster centers that minimize the sum of squared distances from each data point to its closest cluster center. It is well known that the K-means algorithm is prone to outliers and very sensitive to initialization. A natural question to ask is: with

better K-means centers, could we further improve the classification performance? To resolve this, many models have been proposed. Kmeans++[7] initializes the centers in an incremental fashion: a point x' is chosen as the k-th center with probability $y_{k-1}(x')/\sum_{x\in X} y_{k-1}(x)$, where $y^{k-1}(x)$ is the minimal squared distance of x to the current $k-1$ centers. Since the farthest point still has high probability, it continues to suffer from outliers. Tseng [69] presents a weighted K-means algorithm to penalize the outlying objects of a cluster, however the weights are manually designed based on some human-provided priors.

The Generalized Extreme Value (GEV) distribution [22] is a family of probability distributions based on extreme value theory. It unifies three types of extreme value distribution – Gumbel, Frechet and Weibull – each of which is a limiting case for different types of underlying distribution. GEV is often used to model the largest/smallest value from a block of observations, as the block size becomes large. Fernando et al. [28] use GEV to find the best threshold to binarize text documents and scene objects, Scheirer et al. [58] uses GEV for merging classification scores of multiple classifiers in visual object recognition. Burghouts et al. [19] uses Weibull distributions to model the distances from one reference feature vector to other vectors.

In this chapter, an improved K-means clustering is proposed based on the GEV distribution, which performs better with outliers – the learned centers maintain diversity and also reduce unnecessary redundancy. We first cast the K-means algorithm to a block extrema optimization problem. Given K cluster centers, the **minimal squared distance**(MSD) of a given point x to all K centers, $y(x) = \min_k\{||x - c_k||^2\}$, is a block minimum among the blocks of size K. Then the K-means algorithm essentially tries to minimize the average MSD over all training points. With large enough K, we found MSD empirically adheres to Type II Generalized Extreme Value (GEV) distribution. Given a point, its GEV probability of MSD essentially tells how likely this MSD occurs based on the training set. Those points with very low GEV probabilities, either very close to the current chosen centers (very small MSD) or high-probability outliers (with very large MSD), are ignored when we minimize the overall reconstruction errors. Instead, we focus on those points with large GEV probabilities, which are potentially more relevant to the underlying data distribution in high-dimensional space. Therefore, our algorithm has less sensitivity to outliers.

From the algorithmic point of view, our method differs from the K-means algorithm in the way it updates centers. Given a set of points within a cluster, instead of uniformly averaging them to get the new center as K-means, we downgrade all the points with too low/high distance to the centers. Essentially we move the center to the most probable position such that the new center is more likely in the sense of GEV distribution. By doing so, we maintain the diversity and remove the unnecessary redundancy among the cluster centers.

The remaining sections are organized as follows: in Section 5.2 we briefly introduce the Generalized Extreme Value distribution (GEV); in Section 5.3 our GEV-Kmeans clustering algorithm is presented; the experimental results, conclusion and future works are presented in Section 5.4 and 5.5, respectively.

5.2 Generalized Extreme Value Distribution

The generalized extreme value (GEV) distribution [22] is a family of continuous probability distributions based on extreme value theory. It unifies three types of extreme value distribution families, Gumbel, Frchet and Weibull. By the extreme value theorem the GEV distribution is the limit distribution of properly normalized extrema of a block of independent and identically distributed (i.i.d.) random variables.

The formal definition of GEV follows: given a random variable X, we observe its i.i.d. finite sample: $\{x_1, x_2, \ldots, x_K\}$, then the block extreme value $Y = \min(X_1, \ldots, X_K)$ approximately follows the Generalized Extreme Value (GEV) distribution. The Probability Density Function (PDF) is given by

$$\begin{aligned} f(x;\mu,\sigma,\xi) =& \frac{1}{\sigma}\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{(-1/\xi)-1} \times \\ & \exp\left\{-\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{-1/\xi}\right\} \end{aligned} \tag{5.1}$$

for $1+\xi(x-\mu)/\sigma > 0$. (μ, σ, ξ) are location, scale and shape parameters, respectively. Depending on the value of ξ, GEV is equivalent to Gumbel, Frchet and Weibull distributions. Fig 5.1 shows the density function for these three types of distribution family. In particular, the Type II GEV distribution, asymmetric and heavy tailed on the right-hand side, will be used in this chapter.

Some basic statistics measures are given as:

$$\text{mean} = \begin{cases} \mu + \sigma\frac{\Gamma(1-\xi)-1}{\xi} & \text{if } \xi \neq 0, \xi < 1 \\ \mu + \sigma\gamma, & \text{if } \xi = 0 \\ \infty & \text{if } \xi \geq 1 \end{cases} \tag{5.2}$$

and

$$\text{variance} = \begin{cases} \sigma^2(g_2 - g_1^2) & \text{if } \xi \neq 0, \xi < 1/2 \\ \sigma^2\pi^2/6, & \text{if } \xi = 0 \\ \infty & \text{if } \xi \geq 1/2 \end{cases} \tag{5.3}$$

where Γ and γ are the Gamma function and Euler's constant, respectively, and $g_k = \tau(1-k\xi)$.

The GEV distribution has been widely used in modeling block extrema problems. [28] uses GEV to find the best threshold for binarizing text documents and scene objects, while [58] uses it for merging classification scores from multiple classifiers for visual object recognition.

Many problems in machine learning and computer vision can be cast as block-extrema problems. For example, in an Ordinary Least Squares problem, $\beta^* = \text{argmin}_\beta ||x - D\beta||^2$ (where D is the design matrix), the optimal β can be acquired by finding the one which gives the smallest reconstruction errors, within a big block: $\{||x - D\beta_1||^2, \ldots, ||x - D\beta_K||^2\}$. Each β_i could be a random sample from some distribution. Here, we apply GEV theory to improve the K-means clustering algorithm.

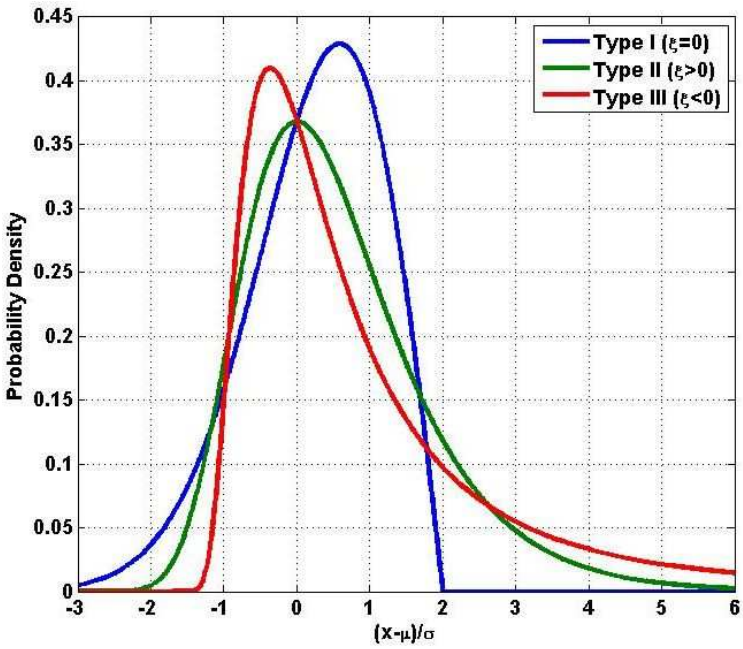


Figure 5.1: PDF for three types of GEV distribution [22].

5.3 GEV-Kmeans clustering

In this section, we present the setup of our GEV-K-means algorithm, to be followed by the optimization and analysis. Here, we use C_k for the k-th cluster, or the k-th cluster centers, interchangably (the context should clarify). Also for brevity, we use **minimal squared distance** and **MSD** equally.

5.3.1 Setup

Given a set of observations $\{x_1, x_2, \ldots, x_N\}$, where $x_i \in \mathbb{R}^d$, K-means clustering aims to partition the N observations into K sets $C = \{C_1, C_2, \ldots, C_K\}$ so as to minimize the following objective function:

$$\min_{\{c_k\}} \quad J = \frac{1}{N} \sum_i \min_k \{||x_i - c_k||^2\} \equiv \frac{1}{N} \sum_i y_i \tag{5.4}$$

here y_i is the **minimal squared distance** of x_i to all centers:

$$y_i = \min_k \left\{||x_i - c_1||^2, \ldots, ||x_i - c_K||^2\right\} \tag{5.5}$$

Observe that y_i is actually a block-min among all K squared distances. To connect Eqn 5.5 with the GEV distribution, we first define the following random variable:

$$D = ||X - CL||^2$$

where $L \in \mathbb{R}^K$ is a random variable, only one of its component is 1, indicating which cluster X is in. By randomly sampling L K' times (K' could be different from K), we further define a new random variable:

$$Y = \min_{\text{all } K' \text{ samples}} \{||X - CL_1||^2, \ldots, ||X - CL_{K'}||^2\} \tag{5.6}$$

Notice each squared distance $||X - DL_k||^2$ is an i.i.d. sample of our random variable D, and also follows the $\mathbb{N}(0, \sigma)^2$ distribution with heavy tail. Since we are only interested in the block-min, we can model the distribution of Y as a Type II Generalized Extreme Value (GEV) distribution.[1]

Knowing the distribution of Y, we can reformulate the K-means clustering as the following optimization problem:

$$\begin{aligned} \min_{\{c_k\}} \quad & J = \mathbb{E}_{P(Y)} Y \\ \text{s.t.: } & \{\mu^*, \sigma^*, \xi^*\} = \text{argmax} \sum_{i=1}^{N} \log P(y_i|\mu, \sigma, \xi) \\ & 1 + \xi^*(y_i - \mu^*)/\sigma^* > 0, \quad \forall i \end{aligned} \tag{5.8}$$

where $P(y|\mu, \sigma, \xi)$ is a GEV distribution as defined in Section 5.2, $E_P(Y)$ is the expectation of Y with respect to the distribution of $P(Y)$. The second constraint comes from the requirement of the GEV distribution in Section 5.2. That is, we first fit the GEV distibution based on the observed training data, and then minimize the expected **minimal squared distance** with respect to this distribution. We know already that $\mathbb{E}_{P(Y)} Y = \mu + \sigma \frac{\tau(1-\xi)-1}{\xi}$ for Type II GEV distributions.

In this chapter, we make the following simplification: instead of sampling multiple times to construct the GEV variable of Y using Eqn 5.6, we use the determinstic process as in Eqn 5.5. Although the assumption required by the GEV distribution is not satisfied here, we can still well approximate it with a GEV distribution in practice (see experimental section for verification). Therefore, the final optimization problem we end up with is the following:

$$\begin{aligned} \min_{\mu, \sigma, \xi, \{c_k\}} \quad & J = \alpha \left(\frac{1}{N} \sum_{i=1}^{N} y_i \right) - \frac{1}{N} \sum_{i=1}^{N} \log P(y_i|\mu, \sigma, \xi) \\ & \text{s.t.: } 1 + \xi(y_i - \mu)/\sigma > 0, \quad \forall i \end{aligned} \tag{5.9}$$

[1] Also, the above sampling process is similar to the generative model of conventional K-means clustering. Randomly sampling L_k is equivalent to randomly sampling from the prior of clusters, i.e., $P(C)$. This prior could be flat(uniform) or follow some other prior distribution. Each cluster can be an additive Gaussian model, and all clusters share the same Gaussian noise variance.

$$x = c_k + \epsilon \tag{5.7}$$

where we use the empirical expectation of $\hat{\mathbb{E}}Y$ instead of $\mathbb{E}_P Y$, and α is a penalty term which balances the two items above. It turns out that this simplification leads to a closed-form updating rule for centers. Given a point x, $P(y)$ simply measures how likely, or how often, its MSD of y could occur; this differs from the probability that x should be assigned to a cluster.

Notice the difference from the conventional K-means algorithm which does not have the second term. Equivalently, the K-means algorithm will treat the distribution of Y as uniform.

Our objective function in Eqn 5.9 balances the reconstruction errors and their likelihood of GEV probabilities. Equivalently, for a given point, the reconstruction error is inversely-proportional weighted by the GEV probability of its MSD. Points with very small P(y) value, or equivalently having too small/big MSD distance (rare events), are basically ignored when we collect the reconstruction errors (since the very large value of its log-likelihood will overshadow the reconstruction error). Instead, we focus on the majority of other points with reasonable P(y) values, which are more relevant to the underlying data distribution.

Let's further illustrate this by considering the following scenario: Given current centers, for those points with very low reconstruction errors but low P(y) values, K-means will be happy to accept the current centers since the goal of K-means is to reduce the reconstruction error. However GEV-kmeans further checks the P(y) value. Lower P(y) values could be caused by two situations: 1) the MSD of those points are very small – in this case, those points are very close to the current chosen centers and they can be explained very well using current centers, which are "uninteresting" for our problem; and 2) the MSD of those points are very large, that is, those points are extremely likely to be outliers with respect to the current centers. GEV-kmeans will reorganize the centers to avoid this situation until we find more reasonable cluster centers.

Our algorithm and its detailed analysis will be presented in sections 5.3.2 and 5.3.3, respectively.

5.3.2 Optimization and algorithm

To solve for the optimization problem in Eqn 5.9, an EM-style algorithm is employed, meaning: we first solve for the GEV parameters (μ^*, σ^*, ξ^*) by fixing centers; and then update centers by fixing the GEV parameters.

For the first part, we use the Matlab function "gevfit" to find the Maximum Likelihood Estimation (MLE) of the GEV parameters, which automatically takes care of the constraints. To update the centers, we take the first-order gradient of J with respect to c_k:

$$\frac{\partial J}{\partial c_k} = \frac{1}{N} \sum_{i \in k} (\alpha + A_i + B_i)(c_k - x_i) \tag{5.10}$$

Algorithm 6 GEV-Clustering

Initialize $\{c_k\}$;
(Optional) Run K-means algorithm several steps;
repeat
Calculate **minimum squared distance** $\{y_i\}$ using Eqn 5.6, and fit the GEV model using $\{y_i\}$ to get (μ, σ, ξ);
update centers $\{c_k\}$ using Eqn 5.11;
until Converge

and setting to zero, provides the following update rule:

$$c_k^* = \sum_{i \in k} \left(\underbrace{\frac{\alpha + A_i + B_i}{\sum_{j \in k}(\alpha + A_j + B_j)}}_{w_i} \right) x_i \tag{5.11}$$

where

$$A_i = \frac{1+\xi}{\sigma} \frac{1}{1 + \xi \frac{y_i - \mu}{\sigma}}, \quad B_i = \frac{-1}{\sigma} \left[1 + \xi \frac{y_i - \mu}{\sigma} \right]^{-1/\xi - 1}$$

and $i \in k$ means all points have minimal Euclidean distance with respect to the k-th cluster.

The algorithm is summarized as follows in Algorithm 6. Similar to [42], it is easy to show that our algorithm converges: in the first step of Algorithm 6, we minimize the negative log-likelihood based on the current centers; and in the second step, we further minimize the objective function by updating centers. Empirically, a fast convergence speed is observed, usually within about 20 iterations.

5.3.3 Analysis

From an algorithmic point of view, our method differs from Kmeans only in how we update the centers, and we still use the minimal distance as a measure to label each point with the nearest center. As in Eqn 5.11, the new center is a linear combination of the points within this cluster, the weights w depend on the values of $A + B$, whose shape is shown in Fig 5.2. We can see that: $A + B$ reaches its peak at some MSD, and is negative if the MSD is smaller than some value, decreasing slower when MSD increases, and never dropping below zero. One interesting consequence is that if most points within a cluster have very small MSD, then the sum of $\sum_{i \in k}(\alpha + A_i + B_i)$ in Eqn 5.11 will be negative.

To see how the weights of w_i affect the updating of centers, we pick two clusters, visualize the image patches within that center and their weights using Eqn 5.11 (see Fig 5.3-Top and Bottom, respectively). Fig 5.3-Top shows a very typical cluster during our iterations. We can see that weights are maximum at some MSD distance, and drop quickly on lower

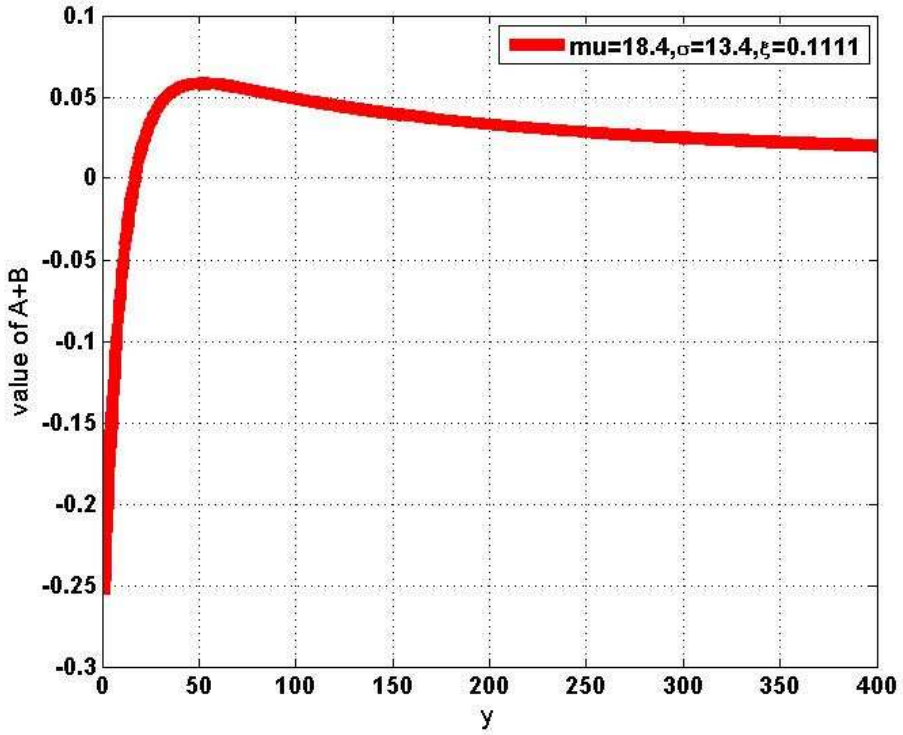
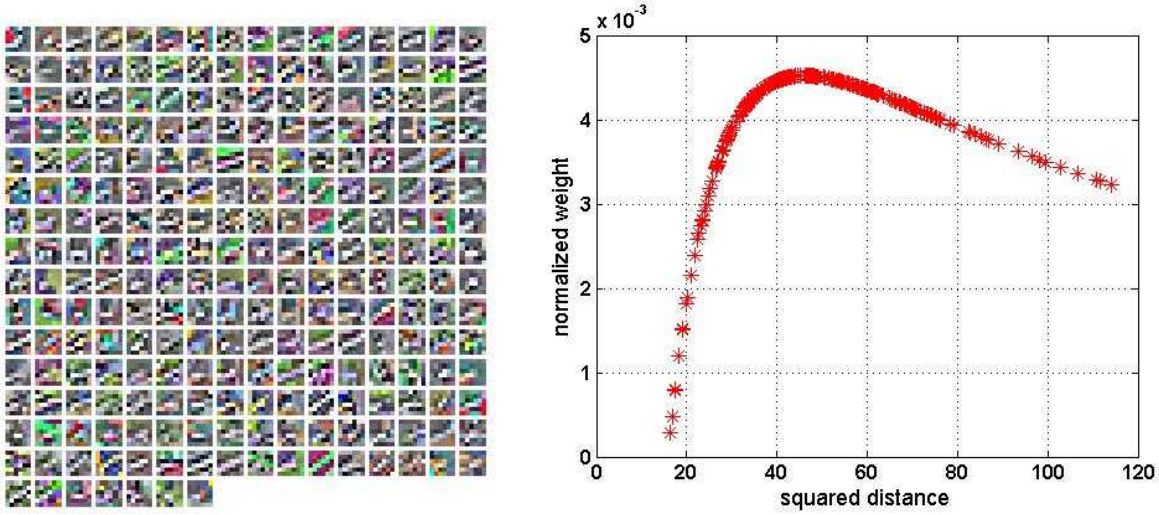


Figure 5.2: Value of A+B, with GEV parameters: $\mu = 18.4, \sigma = 13.4, \xi = 0.1111$

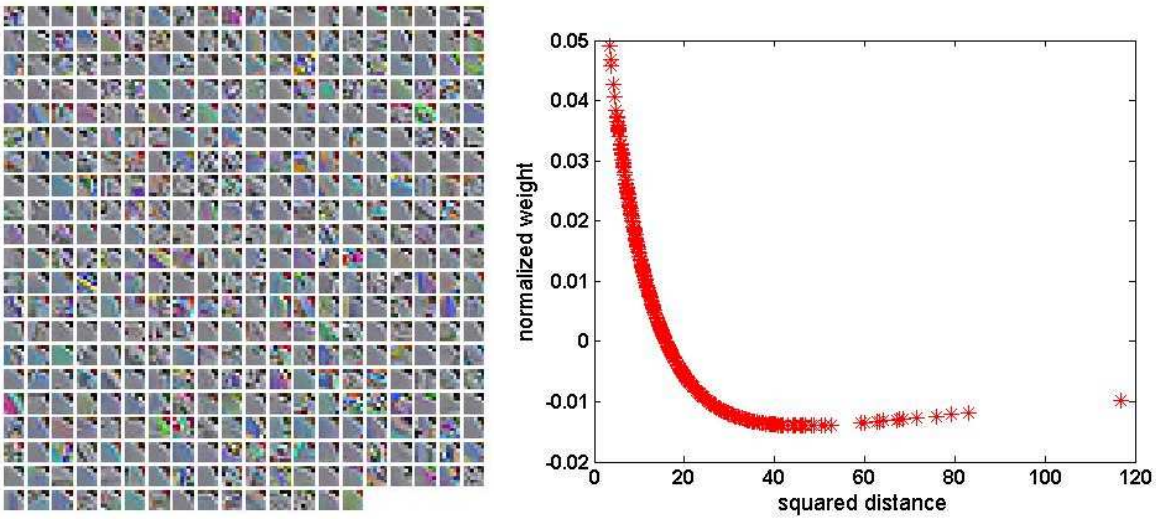
MSD values and more slowly on larger MSDs. That is, when we update the center for this cluster, we favor those with reasonable MSD and simultaneously downgrade points with larger/smaller MSDs. This brings two benefits: a) if some points have MSD greater than the optimal distance, meaning they are potentially outliers, we will not consider them equally with points of optimal distance, but downgrade them; b) those points with MSD smaller than the optimal distance will also be downgraded. In other words, after linear averaging, we move our center to a new position such that all of the points within this cluster have larger $P(y)$, i.e., are more probable. Doing this can effectively remove outliers.

More interestingly, Fig 5.3-Bottom visualizes a cluster where $\sum(A + B + \alpha) < 0$. We can clearly see that we will prefer points with very large/very small MSD distance, rather than optimal distance. The consequence is that the updated center will be considerably off the current center in order to reduce the objective function. It is very likely that this cluster (visually speaking, for example, gray image patches with right-top corner having a dot) could be a dangling outlier. Consequenctly, this center will be deleted at the next iteration, and all of its associated points will be reclustered into other centers. During this process, the objective function will be temporarily increased. Empirically, many cluster centers are deleted during the first few iterations, with most arising from a negative sum-of-weights in

Eqn 5.11.



(a) Cluster 1 – Left:image patches; Right: weights w



(b) Cluster 2 – left:image patches; Right: weights w

Figure 5.3: image patches within each cluster and their weights w

Conceptually, Fig 5.3-Top and Bottom corresponds to the scenarios shown in Fig 5.4-Left and Right, respectively. Suppose we have three data point clouds as shown in Fig 5.4, with PointSet 3 being outliers.

(a) If we already have the centers for PointSet 1 and PointSet 2 (see Fig 5.4-Left), how do we update these centers with the presence of PointSet 3? Since the MSDs for all

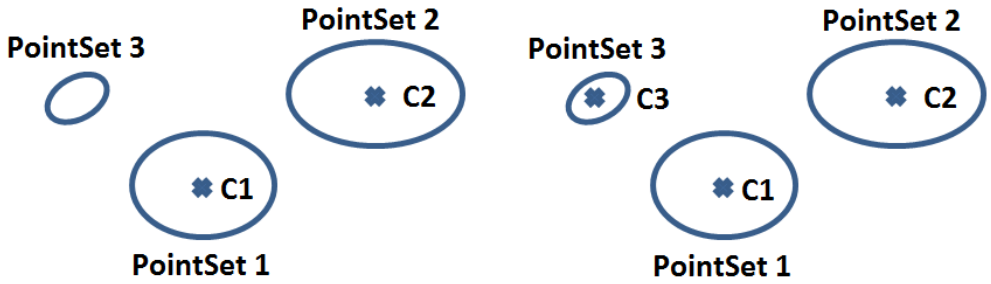


Figure 5.4: Simple example. Left: we have known C_1/C_2; Right: we have known three centers $C_1/C_2/C_3$.

points in PointSet 3 are very high (with low P(y) values), we will update the first center (only) by discounting all points in 3. That is, a dangling point set, possibly being outliers, will not significantly influence our centers;

(b) Now, consider if we already have three clusters for our dataset. Since the PointSet 3 are dangling outliers, we would like to get rid of its corresponding center. Most of the points in PointSet 3 have very small **minimum squared distance**, therefore the corresponding $A + B$ will be negative. Consequently, the center C_3 will be removed at the next iteration.

Notice our algorithm can effectively delete inappropriate centers without adding new ones. However, the number of centers will stabilize quickly. At the end of all iterations, the new centers will have the following properties: a) the MSD for points within each cluster will have higher probability, being neither too large nor too small; and b) the reconstruction error will be very small.

5.3.4 Initialize K-means Clustering

From the above analysis, a new initialization method is proposed for K-means based on the GEV distribution. Comparing with the conventional initialization methods, our method, considering the data distribution property, will give better reconstruction errors. The algorithm is summarized in Algorithm 7.

5.4 Experimental results

In this section, we first conduct experiments to demonstrate the GEV distribution can help find better cluster centers and then conduct experiments to verify the proposed GEV-Kmeans clustering algorithm on the CIFAR-10 visual object recognition image benchmark database [67].

Algorithm 7 Initialize K-means using GEV

Set cluster centers C to be empty;
Randomly choosing a sample x and adding to C;
repeat
Calculate **minimum squared distance** $\{y_i\}$ using Eqn 5.6, and fit the GEV model using $\{y_i\}$ to get (μ, σ, ξ);
Randomly sampling a value v from $GEV(\mu, \sigma, \xi)$, and choosing the correspoinding sample into C
until K centers are chosen

data/K	10	20	30	50	100
Cloud 1	8.8	18.8	24.8	34.8	45.9
Cloud 2	-2.7	2.9	11.3	22.5	37.3
image_seg.	1.0	-5.4	5.3	13.3	31.0
Lenna	45.8	81.0	85.4	93.3	/

Table 5.1: Improvement factor of reconstruction error

5.4.1 UCI dataset

Cloud 1, **Cloud 2**, **image segmentation** and **Lenna** are popular clustering datasets [1]. In this section, we use this dataset to demonstrate our initialization method for K-means clustering. For K chosen cluster centers, the reconstruction/quantization error is calculated as:

$$Err = \sum_{i=1}^{N} \min_{k} \{||x_i - c_k||\}$$

We initialize the K-means using our method, and then run K-means clustering. Then we calculate the improvement ratio of the reconstruction error with respect to the error of Kmeans with random initialization. Ten runs are conducted for each K, we report the average improvement factor. The results are shown in Table 5.1:

5.4.2 CIFAR-10

The CIFAR-10 dataset [67] is a benchmark image database for the visual object recognition task. It contains 50K/10K training/testing color images, each with size 32×32. Ten categories are considered, including car, plane, etc.

Here, to compare the capability of our GEV-Kmeans algorithm against the conventional K-means algorithm, we exactly follow the experimental protocol of [2]. 1) 400K 6×6 random patches are extracted from the training images; 2) preprocessing: each local image patch undergoes contrast normalization, and whitening is applied on all image patches; 3) using the processed images patches, we perform GEV-Kmeans clustering, and cluster centroids are

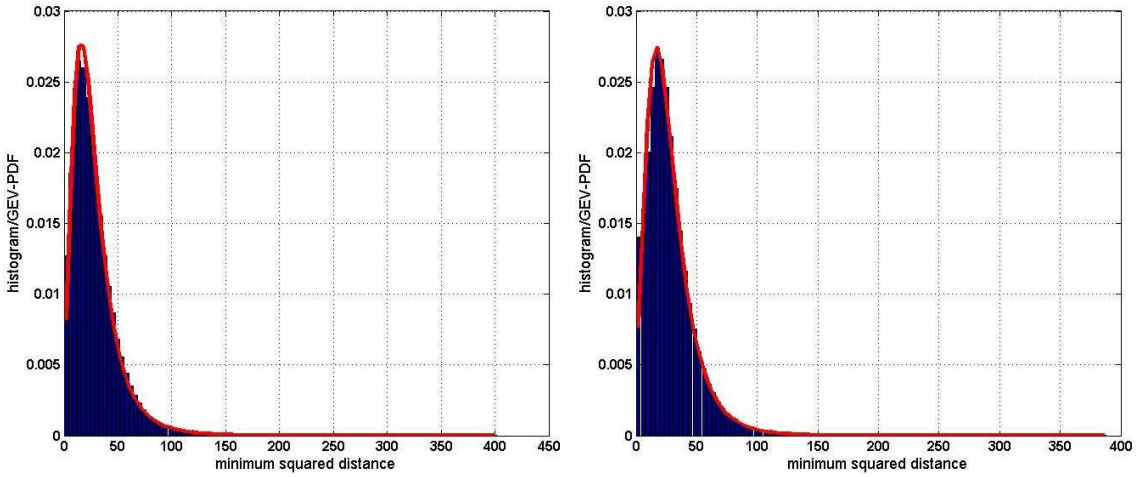


Figure 5.5: Histogram of **minimum squared distance** and the estimated Probability Density Function of GEV distribution. Left: the centers are initialized randomly; Right: after 100 iterations.

learned. After learning the centers, each 6×6 patch can be mapped into a feature vector $z \in \mathbb{R}^K$: $z_k = ||x - c_k||_2$, further we threshold z to get the final sparse feature representation for this patch by: $f_k(x) = \max\{0, \mu(z) - z_k\}$, where $\mu(z)$ is the average of vector z (see [2] for details); 4) each 32×32 image is divided into 6×6 patches with stride equal to 1, and, with average pooling on a 2×2 grid, the final vectors are concatenated into a long vector ($\mathbb{R}^{4K}$); 5) linear SVM is applied on these vectors for classification.

Here, we initialize the GEV-kmeans with five steps of the K-means algorithm, with the K-means' centers initialized randomly.

GEV assumption on MSD

We argue that the **minimum squared distance** satisfies a GEV distribution. To verify this assumption, we apply our GEV-Kmeans clustering algorithm to the pre-processed image patches. Fig 5.5-Left/Right shows the acquired normalized histogram overlayed with the estimated Probability Density Probabilities (PDF), at iteration 1 and 100, respectively. Since our algorithm will remove some of the clusters, the initial K we set is 1300, and the final number of GEV-Kmeans centers reaches 1210.

First we validate our assumption by visual inspection. From Fig 5.5, we can see that the normalized histogram of MSD and the fitted GEV Probability Density Function match quite well, visually. As a simple goodness-of-fit measure, we compare the sample mean $\sum_i y_i/N$ and expectation ($EY = \mu + \sigma\frac{\tau(1-\xi)-1}{\xi}$), empirical standard deviation and the standard deviation $\sigma\sqrt{g_2 - g_1^2}$ at iteration 1 and 100, respectively (see Table 5.2). The parameters for the GEV distribution, (μ, σ, ξ), are estimated for each iteration. We can see that the

Iter.	sample mean	$\mathbb{E}Y$	sample std	$std(Y)$
1	27.9909	28.1062	21.0654	22.423
100	27.8237	27.8520	20.089	20.4640

Table 5.2: Validation of GEV assumption

numbers match well. In other words, the assumption that the MSD's distribution can be approximated by a GEV distribution appears valid, in practice.

Visualization of learned centers

Before we present classification results, we visualize the cluster centers we learned using our GEV-Kmeans in Fig 5.6. Similar to K-means and other methods such as sparse coding, the K-means algorithm, etc, the learned centers resemble Gabor-like filters. Notice that our centers are sorted based on their standard deviation.

Classification Performance

After we learned the cluster centers using GEV-Kmeans, we follow the experimental protocol in [2] for classification. Again, we use five steps of K-means to initialize our centers, and performed 25 iterations of GEV-Kmeans. As with the K-means algorithm, GEV-Kmeans is sensitive to the initialization. In this chapter, we run GEV-Kmeans five times and report the one giving the best classification accuracy. In order to train the linear SVM, we fine tune the penalty parameter C in the SVM on one set of experiments for each K, and use it for all five runs.

Also, since the GEV-Kmeans clustering algorithm could delete centers during the early iterations, and the number of centers stabilizes quickly, the number of centers is not exactly the same as [2] used. The actual number of centers of our results are: [108, 408, 811, 1224, 1630, 4020]. To fairly compare with the K-means result in [2], we run the algorithm in [2] three times for $K = 100, 400, 800, 1200$ separately, and choose the one which gives the best accuracy. Since we don't fine tune the linear SVM parameters C for those experiments, the results shown in the figure are slightly poorer than those reported in [2].

The classification accuracy on the CIFAR-10 dataset using different number of centers is summarized in Fig 5.7.

From the figure, we have the following observations:

(a) For all K, GEV-kmeans outperforms K-means, which demonstrates better cluster centers can be achieved using our GEV-kmeans algorithm;

(b) When $K = 1224$, our algorithm (78.10%) outperforms the K-means algorithm with K=1600 centers, 77.9%.

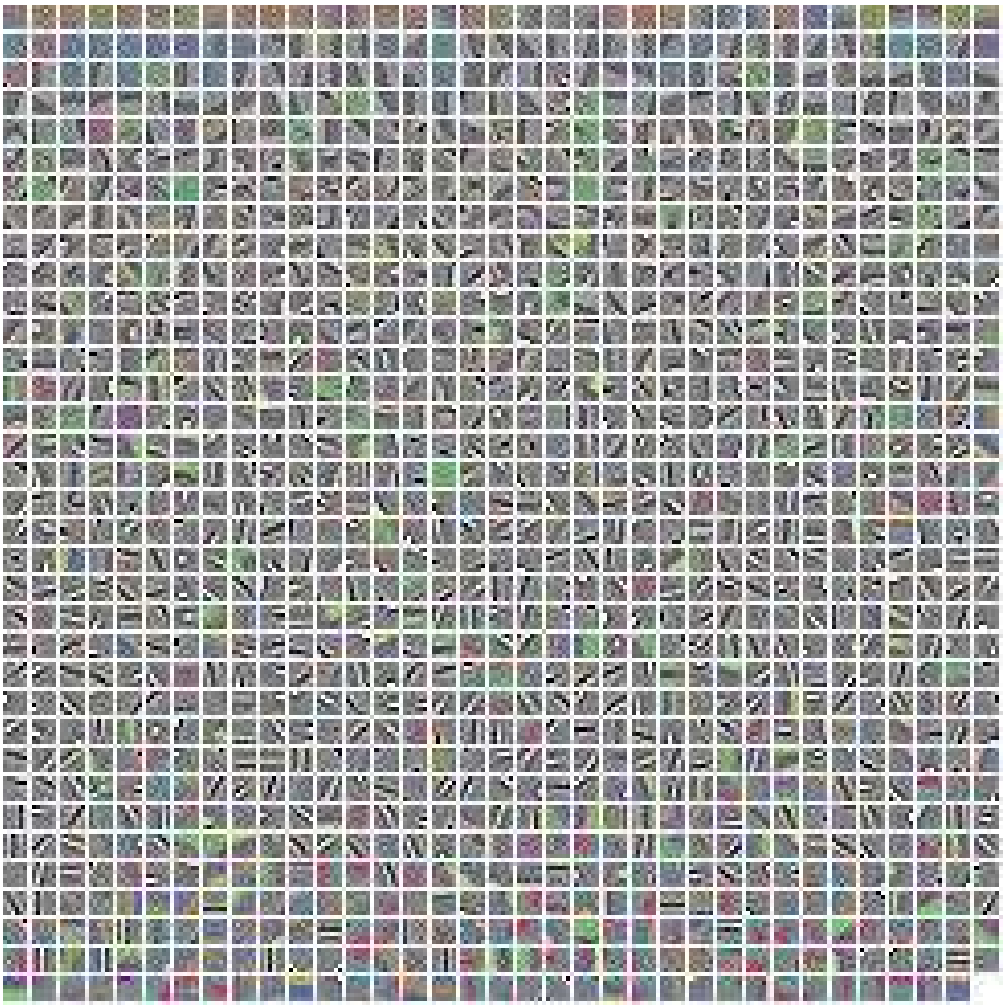


Figure 5.6: Visualization of learned clustering centers. The centers are sorted by their standard deviation, from low to high.

(c) We can gain more when the number of centers is moderately large. One possible reason is that the GEV distribution requires a relatively large number of i.i.d. samples when taking the block maxima/minima. Therefore, the GEV approximation is more suitable with this range of number of centers. When the number of centers reaches much higher, the gain of GEV-Kmeans decreases. Our classification accuracy is 79.77%, while K-means in [2] can reach 79.6%. There is no longer a need to carefully choose good centers in this situation. This is also consistent with the observations of Lee et al. [2].

The best classification accuracy on the CIFAR-10 dataset, 81.5%, is achieved using Orthogonal Matching Pursuit (OMP-1) (with dictionary size equal to 6000) as feature extractor for each patch and "soft threshold" as non-linearity layer on top of the OMP-1 codes [21].

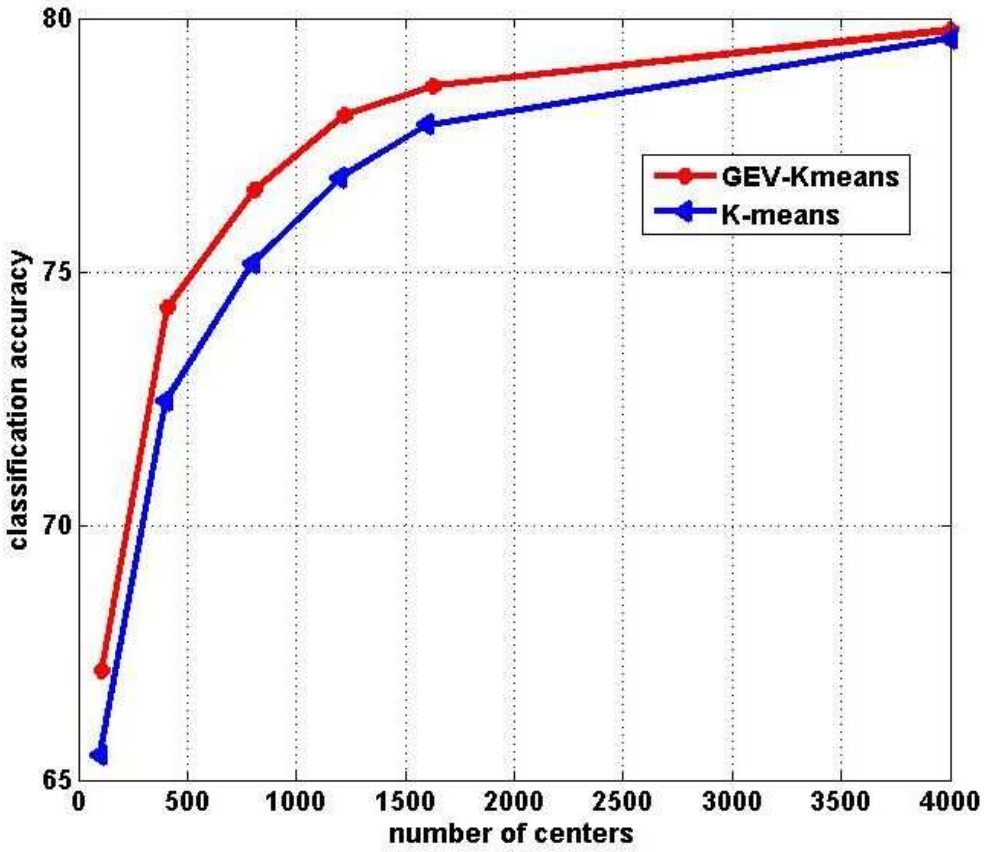


Figure 5.7: Classification accuracy using different number of centers.

Although our algorithm cannot achieve the state-of-the-art results on the CIFAR-10 dataset, we experimentally demonstrate our GEV-kmeans algorithm can learn better features than K-means clustering algorithm. One possible reason is that we can handle outliers well.

Reconstruction Error

Finally, we compare the reconstruction error:

$$\text{Recon. error} = \frac{1}{N}\sum_{i=1}^{N}\min_{k}||x - c_k||^2$$

between GEV-kmeans and K-means algorithm, see Fig 5.8. The cluster centers learned from GEV-kmeans cause slightly greater reconstruction errors than K-means. However, our goal in Eqn 5.9 is not to reduce this error, but to get better cluster centers.

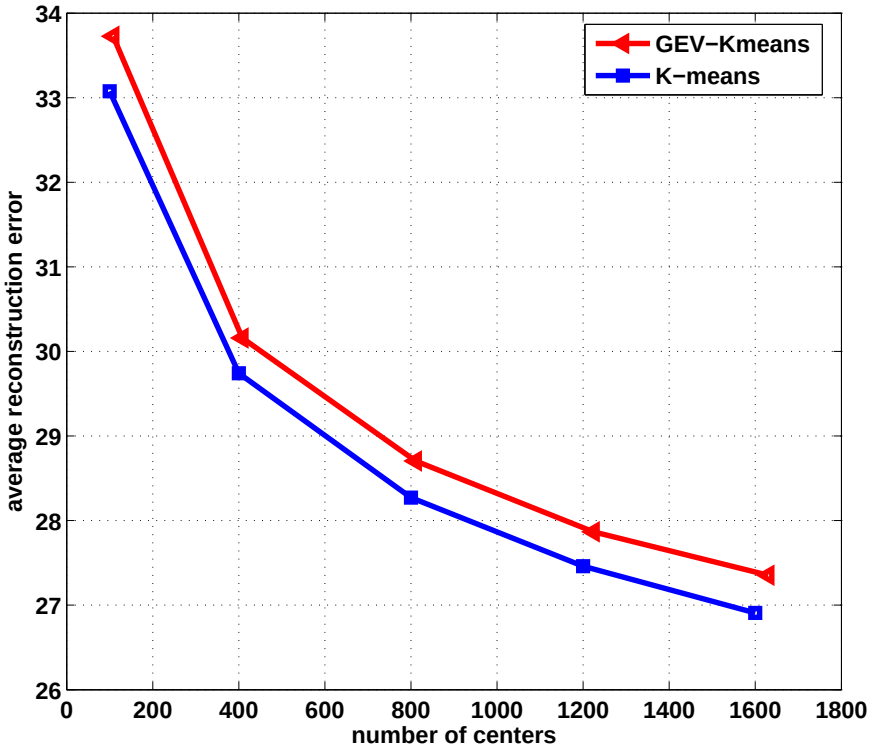


Figure 5.8: Reconstruction Errors of GEV-kmeans and K-means

5.5 Conclusion and future work

An improved K-means clustering algorithm based on GEV distribution, GEV-Kmeans, has been proposed to learn better features. It outperforms the K-means algorithm when K is moderately large. We first cast K-means clustering as a block extrema optimization problem, and approximate the distribution of the **minimal squared distance** using the GEV distribution. With this data-dependent global information, our learned centers are less sensitive to outliers, with computational overhead comparable with the K-means algorithm. Experiments show that our algorithm works better with a moderate number of cluster centers. Future work, could apply the GEV-kmeans algorithm with other non-linearity layer and pooling layers as suggested in Coates et al. [21] and also extend the approach to sparse coding.

Chapter 6

Conclusions and future works

In this dissertation we have explored the image-patch modeling problem by considering image data and scene geometric information in an integrated manner. A geometry-aware sparse-coding strategy is presented to efficiently encode an image patch while remaining respectful of the geometric constraints embedded in a light-field. Two applications based on this model are applied: joint image denoising and angular/spatial super-resolution, in a light-field. Experimental results show the superiority of the developed method over its counterparts – image modeling based on a single image.

Additionally, two efficient image patch modeling methods are proposed, based on Kernel Principle Analysis and the Generalized Extreme Distribution. Both of these methods show better performance than others in a number of challenge classification tasks. As a modeling tool for EPI-patches, our next step is to further improve performance using these two image-patch modeling methods.

Light-field data provides a new type of image information, rich in redundancy and expressive of the geometric structure of a scene. We expect that exploiting the redundant information embedded in a light-field will lead to discovery of other benefits relevant to computer vision challenges beyond those we focussed upon in this thesis.

Appendix A

2D grid camera array calibration

The calibration of a 2D camera array, see Fig 1.1 in Chapter 1, is essential for our light-field data application. In this chapter, our calibration scheme is presented and evaluated.

A.1 Setup

The goal of our calibration is to guarantee the following criteria: 1) Each row of cameras only has vertical disparity (see Eqn 2.7), and 2) Each column of cameras only have horizontal disparity (se Eqn 2.8). See Fig 2.1 in Chapter 2.

Some previous works have been presented along this line previously [9][62][10][41][37]. [70] uses plane + parallax idea for calibration a 2D grid cameras. [41] presents a work to rectify a 1D linear camera array. During this chapter, we divide the calibration process into two steps: 1) geometrical calibration, finding the internal and external parameters of all cameras with respect to a chosen reference camera, and 2) the rectification process, which guarantees the horizontal and vertical disparity disappearing for each row and column, respectively. The second step is cast into a non-linear optimization problem, finding a homography matrix for each camera with respect to the reference camera such that certain alignment goals are met.

Similar to the other camera calibration systems, we use a checker board as our calibration target, where we know the checker size and have carefully measured their colors (used to also adjust colors across all 2D cameras).

A.2 Geometric calibration

Since the cameras share the same overlap of the checker-board, it is much easier to find the corresponding checker corners across all 2D images. After finding the corresponding points, we minimize the re-projection errors of all points, in order to get the geometrical parameters of all cameras. With respect to a chosen reference camera, each pinhole camera

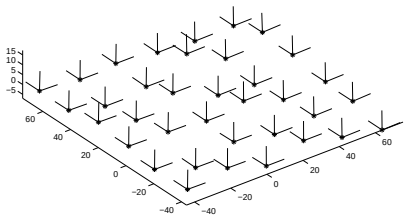


Figure A.1: Optimized camera centers after Geometrical Calibration.

is charactered by the following parameters:

$$\{K_{ij}, R_{ij}, T_{ij}\}, \quad i \in \{1, \ldots, R\}, \quad j \in \{1, \ldots, C\} \tag{A.1}$$

After this stage, the optimized camera centers layout after the geometric calibration is shown in Fig A.1. It can be shown that all cameras' centers are approximately on a plane.

We also calculate the horizontal baseline vector $\mathbf{b^{(h)}} \in \mathbb{R}^C$ (see Eqn 2.6 in Chapter 2) by averaging the baseline of (i, j)-th cameras with respect to the reference camera across all rows. Similarly, we get the vertical baseline vector $\mathbf{b^{(v)}} \in \mathbb{R}^R$ (see Eqn 2.6 in Chapter 2).

The second rectification stage will take these estimated camera parameters and refine them to meet our final goal.

A.3 2D rectification process

Our setup is described as follows: given a calibrated 2D camera array, $\{C_{ij}\}$, where $i = 1, \ldots, R$ and $j = 1, \ldots, C$, we want to find a set of homography matrices, $\{H_{ij}\}$, such that, the warped images after H will be strictly a 2D light field. In this work, we assume H_{ij} will wrap the (i, j) image to the reference camera's image plane, $C_{ref,ref}$.

The basic assumption is our cameras are physically approximately located on a 2D grid. This error is determined or dominated by how far our physical design of cameras is off the ideal 2D grid situations.

Initial guess of $\{H_{ij}\}$

After getting the geometric parameters of all cameras, we can roughly estimate the homography matrices which will be applied to each camera:

$$H_{ij} = K_{ref,ref} R_{ij} (K_{ij})^{-1} \tag{A.2}$$

Nonlinear refinement

The initial guesses of the homography matrices are further refined using a non-linear optimization process.

Given a 3D point P, it projects on the 2D camera array and forms the image points, see Fig 2.1:

$$\{x_{i,j}^{(P)}, y_{i,j}^{(P)}\}, \quad i \in \{1, \ldots, R\}, \quad j \in \{1, \ldots, C\}$$

Due to the special structure of our 2D camera array, we have the following observations:

- ⋆ $\mathbf{X_{slope}}$: For each i-th row, we define a 2D line formed by the x-coordinates of C image points, $\{x_{i,\cdot}\}$, with respect to the horizontal baseline, as $l^h = \{s, u\}$ [1]. Then all lines should be the same across all rows. The slope s in the horizontal EPI-image represents the depth of the 3D point of P (see Eqn 2.9);

- ⋆ $\mathbf{\Delta Y}$: For each i-th row, the vertical disparity will disappear. That is, the y-coordinates of C image points, $\{y_{i,\cdot}\}$, are all equal:

$$y_{ij}^{(P)} - y_{i,j'}^{(P)} = 0, \quad j, j' \in \{1, \ldots, C\}, \quad j! = j' \tag{A.3}$$

- ⋆ $\mathbf{Y_{slope}}$: Similarly, for each j-th column, we define a 2D line formed by the y-coordinates of R image points $\{y_{\cdot,j}\}$, with respect to the vertical baseline, as $l^v = \{t, v\}$. Then all these lines should be the same across all columns. Similarly, the slope t is the depth of P (see Eqn 2.9);

- ⋆ $\mathbf{\Delta X}$: For each j-th column, the horizontal disparity will disappear. That is, the x-coordinates of R image points, $\{x_{\cdot,j}\}$, are all equal:

$$x_{ij}^{(P)} - x_{i',j}^{(P)} = 0, \quad i, i' \in \{1, \ldots, R\}, \quad i! = i' \tag{A.4}$$

- ⋆ Finally, the slope of the horizontal/vertical 2D lines, l^h and l^v, should be the same for all row and column, $s = t$. That is, the depth recovered from each row and each column of 1D camera array should be the same.

Base on the above observations, the following optimization problem is formulated to refine the homography matrices for all cameras, where each part is labeled as the above description:

$$\{H_{ij}^*\} = \underset{\{H_{ij}\}}{\operatorname{argmin}} \left[\sum_{i=1}^{R} \underbrace{\left|\left| x_{i,\cdot} - (s \times \mathbf{b^{(h)}}_i + u) \right|\right|^2}_{\mathbf{X_{slope}}} + \underbrace{||y_{i,\cdot} - \overline{y_{i,\cdot}}||^2}_{\mathbf{\Delta Y}} \right] + \left[\sum_{j=1}^{C} \underbrace{\left|\left| y_{\cdot,j} - (s \times \mathbf{b^{(v)}}_j + v) \right|\right|^2}_{\mathbf{Y_{slope}}} + \underbrace{||x_{\cdot,j} - \overline{x_{\cdot,j}}||^2}_{\mathbf{\Delta X}} \right] \tag{A.5}$$

where $\overline{y_{i,\cdot}}$ and $\overline{x_{\cdot,j}}$ is the mean of $\{y_{i,\cdot}\}$ and $\{x_{\cdot,j}\}$, respectively. $\{s = t, u, v\}$ are not part of the parameters and are estimated in the least-square sense for each point P.

[1] A line is defined as: $l: \quad y = s \times x + u$.

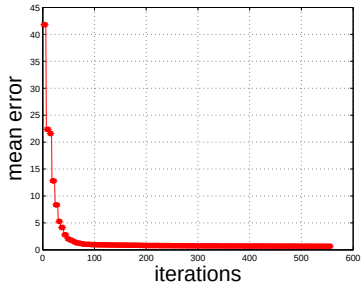


Figure A.2: Optimization converge.

	Ave Error	X-slope	ΔX	Y-slope	ΔY
before	41.81	46.98	20.23	62.64	37.39
after	0.51	0.73	0.60	0.40	0.33

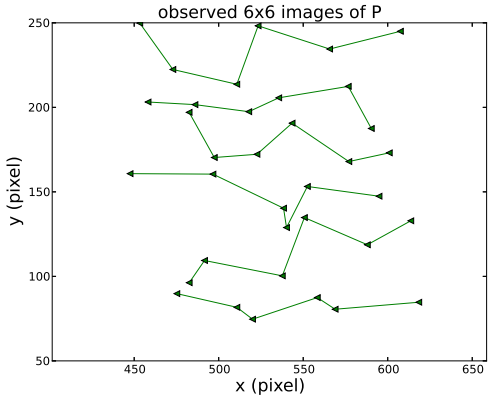
Table A.1: Detailed errors in pixel before and after the optimization.

A.4 Experimental results

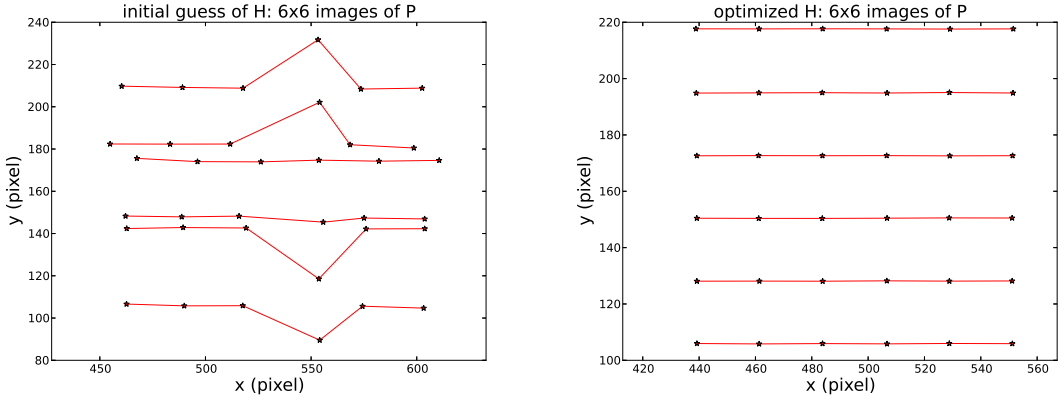
In this section, we test our calibration algorithm on the 6×6 camera array to demonstrate our calibration performance. The objective function converges monotonically and reaches the reasonable solution within roughly 100 iterations, as shown in Fig A.2. Table A.1 shows the detailed error distribution before and after optimization, see Eqn A.5 for the detailed description of each term. It can be seen that our optimization is very effective to reduce the expected errors.

To validate the rectification performance, given a 3D Point, we inspect the projected locations across all 6×6 images before and after our optimization, see Fig A.3. We can see that the optimal homography matrices from our non-linear refinement meet our goal for a light field, see Section A.3.

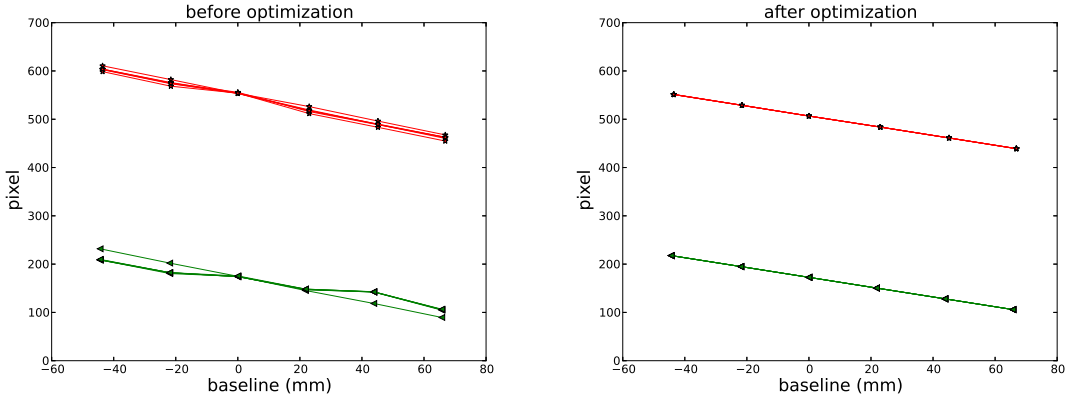
In order to visually inspect the rectification performance, we choose an image point from the middle-view image and also draw the corresponding points across all 6×6 images after optimization, see Fig A.4-top and bottom, respectively. From the figures, we can see that: 1) the vertical disparity disappears for each row of cameras, and 2) the vertical disparity disappears for each column of cameras. Also note the reference points (Top/Bottom in Fig A.4) locate at different depth level, which suggests our algorithm can cover large depth range. Finally, Fig A.5 shows all rectified images.



(a) Observed corresponding locations.



(b) Corresponding points projected by H_0 and H^*.



(c) $\mathbf{X_{slope}}$ / $\mathbf{Y_{slope}}$ projected by H_0 and H^*.

Figure A.3: Optimization results on a point. (a). The observed locations for a 3D point P. (b). From Left to Right: the projected locations using the initial H_0 and the optimized H^*, respectively. (c): $\mathbf{X_{slope}}$ / $\mathbf{Y_{slope}}$ using the initial H_0 and the optimized H^*, respectively.

Figure A.4: Rectification on a point. Top/Bottom: show the corresponding points for a chosen reference image point, labeled as "squared-block", across 6×6 images.

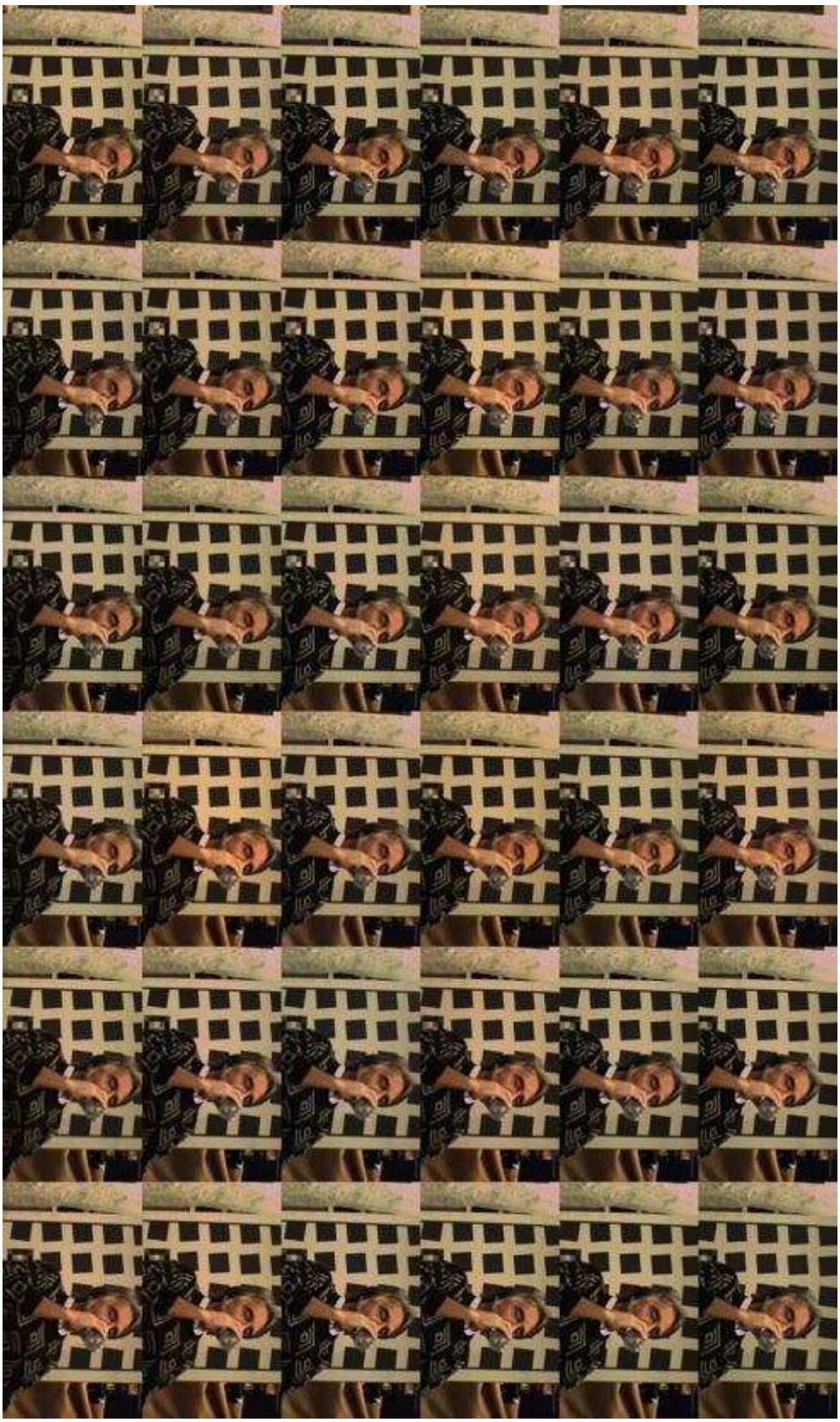


Figure A.5: Rectified images

Bibliography

[1] D.J. Newman A. Asuncion. *UCI Machine Learning Repository*. 2007. URL: `http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html`.

[2] Honglak Lee Adam Coates and Andrew Ng. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *AISTATS* (2011).

[3] Michal Aharon, Michael Elad, and Alfred Bruckstein. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". In: *Structure* (2006).

[4] Carlos Alzate and Johan A. K. Suykens. "Multiway Spectral Clustering with Out-of-Sample Extensions through Weighted Kernel PCA". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 32.2 (2010), pp. 335–347.

[5] The (New) Stanford Light Field Archive. `http://lightfield.stanford.edu`.

[6] Omar Arif. "Robust target localization and segmentation using statistical methods". In: *Ph.D dissertation, Georgia Institute of Technology* (2010).

[7] D Arthur and S Vassilvitskii. "k-means++: The advantages of careful seeding". In: *Proceedings of the eighteenth annual ACMSIAM symposium on Discrete algorithms* (2007).

[8] H. Harlyn Baker and Robert C. Bolles. "Generalizing epipolar-plane image analysis on the spatiotemporal surface". In: *In IJCV* (1989).

[9] Harlyn Baker and Zeyu Li. "Camera and Projector Arrays for Immersive 3D Video". In: *Proceedings of the 2Nd International Conference on Immersive Telecommunications*. IMMERSCOM '09. 2009, 23:1–23:6.

[10] Harlyn Baker et al. "Capture considerations for multiview panoramic cameras". In: *CVPR Workshops*. 2012, pp. 37–44.

[11] Amir Beck and Marc Teboulle. "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems". In: *IEEE Transaction on Image Processing* (2009).

[12] Yoshua Bengio. "Learning deep architectures for AI". In: *Dept. IRO, Universite de Montreal* (2007).

[13] M. Blum et al. "A Learned Feature Descriptor for Object Recognition in RGB-D Data". In: *ICRA* (2012).

[14] L. Bo, X. Ren, and D. Fox. "Depth kernel descriptors for object recognition". In: *IROS* (2011).

[15] L. Bo, X. Ren, and D. Fox. "Unsupervised Feature Learning for RGB-D Based Object Recognition". In: *ISER* (2012).

[16] Robert C. Bolles et al. "Epipolarplane image analysis: An approach to determining structure from motion". In: *IJCV* (1987).

[17] Y. Boureau et al. "Learning mid-level features for recognition". In: *CVPR* (2010).

[18] Antoni Buades and Jean-Michel Morel Bartomeu Coll. "A non-local algorithm for image denoising". In: *CVPR* (2005).

[19] G Burghouts, A Smeulders, and J Geusebroek. "The Distribution Family of Similarity Distances". In: *Advances in Neural Information Processing Systems* 20 (2007). Ed. by J C Platt et al., pp. 1–8.

[20] Peng Cheng, Wanqing Li, and Philip Ogunbona. "Greedy Approximation of Kernel PCA by Minimizing the Mapping Error". In: *Proceedings of the 2009 Digital Image Computing: Techniques and Applications*. DICTA '09. 2009.

[21] Adam Coates, Andrew Y Ng, and Serra Mall. "The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization". In: *ICML* 28.28 (2011). Ed. by Lise Getoor and TobiasEditors Scheffer, pp. 921–928.

[22] Stuart Coles. "An introduction to statistical modeling of extreme values". In: *Springer Series in Statistics Springer-Verlag, London* (2001).

[23] Trevor F. Cox and M.A.A. Cox. *Multidimensional Scaling, Second Edition*. 2000.

[24] K. Dabov et al. "Image denoising by sparse 3d transform-domain collaborative filtering". In: *TIP* (2007).

[25] Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection". In: *International Conference on Computer Vision & Pattern Recognition* (2005).

[26] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. "Fast Monte Carlo Algorithms for Matrices II: Computing a Low-Rank Approximation to a Matrix". In: *SIAM Journal on Computing* 36 (2004), p. 2006.

[27] Michael Elad and Michal Aharon. "Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries". In: *IEEE Transactions on Image Processing* 15.12 (2006), pp. 3736–3745.

[28] Basura Fernando, Sezer Karaoglu, and Alain Trmeau. "Extreme Value Theory Based Text Binarization In Documents and Natural Scenes". In: *International Conf. on Machine Vision* (2010).

[29] Andriy Gelman. "Compression of Multiview Images using a Sparse Layer-based Representation". In: *Ph.D thesis, Imperial College London* (2012).

[30] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose". In: *IEEE Trans. Pattern Anal. Mach. Intelligence* (2001).

[31] David B. Grimes and Rajesh P. N. Rao. "Bilinear Sparse Coding for Invariant Vision". In: *Neural Computation* (2005).

[32] Roger B. Grosse et al. "Shift-Invariance Sparse Coding for Audio Classification". In: *CoRR* (2012).

[33] G E Hinton and R R Salakhutdinov. "Reducing the dimensionality of data with neural networks." In: *Science* 313.5786 (2006), pp. 504–7.

[34] Kevin Jarrett et al. "What is the best multi-stage architecture for object recognition?" In: *2009 IEEE 12th International Conference on Computer Vision* (2009).

[35] Kwang In Kim, Matthias O. Franz, and Bernhard Sch?lkopf. "Iterative kernel principal component analysis for image modeling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005), p. 2005.

[36] Kwang In Kim and Younghee Kwon. "Single-Image Super-Resolution Using Sparse Regression and Natural Image Prior". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2010).

[37] Gregorij Kurillo et al. "Geometric and Color Calibration of Multiview Panoramic Cameras for Life-Size 3D Immersive Video". In: *3DV*. 2013.

[38] K. Lai et al. "A Large-Scale Hierarchical Multi-View RGB-D Object Dataset". In: *ICRA* (2011).

[39] Honglak Lee et al. "Efficient sparse coding algorithms". In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. Platt, and T. Hoffman. Cambridge, MA: MIT Press, 2007, pp. 801–808.

[40] Anat Levin, William T. Freeman, and Frédo Durand. "Understanding Camera Trade-Offs through a Bayesian Analysis of Light Field Projections". In: *ECCV* (2008).

[41] Zeyu Li, Harlyn Baker, and Geogorij Kurillo. "Projective Epipolar Rectification for a Linear Multi-imager Array". In: *3DPVT*. 2010.

[42] S Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* (1982).

[43] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *Int. J. Comput. Vision* 60.2 (Nov. 2004).

[44] Julien Mairal, Guillermo Sapiro, and Michael Elad. "Learning Multiscale Sparse Representations for Image and Video Restoration". In: *Multiscale Modeling & Simulation* (2008).

[45] Julien Mairal, Guillermo Sapiro, and Michael Elad. "Multiscale Sparse Image Representationwith Learned Dictionaries". In: *ICIP (3)*. 2007.

[46] Aleix Martínez and Robert Benavente. *The AR Face Database*. Tech. rep. Computer Vision Center, 1998. URL: `http://www.cat.uab.cat/Public/Publications/1998/MaB1998`.

[47] Kaushik Mitra and Ashok Veeraraghavan. "Light field denoising, light field superresolution and stereo camera based refocussing using a GMM light field patch prior". In: *CVPR Workshops* (2013).

[48] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. 2001.

[49] Ren Ng. "Digital Light Field photography". In: *Ph.D thesis* (2006).

[50] M. Okutomi and T. Kanade. "A Multiple-Baseline Stereo". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 15.4 (Apr. 1993), pp. 353–363. ISSN: 0162-8828. DOI: `10.1109/34.206955`. URL: `http://dx.doi.org/10.1109/34.206955`.

[51] B A Olshausen and D J Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (1996), 607609.

[52] Bruno A. Olshausen and David J. Fieldt. "Sparse coding with an overcomplete basis set: a strategy employed by V1". In: *Vision Research* 37 (1997), pp. 3311–3325.

[53] Vlad Popovici, Samy Bengio, and Jean-Philippe Thiran. "Kernel Matching Pursuit for Large Datasets". In: *Pattern Recognition* 38.12 (2005), pp. 2385–2390.

[54] Vlad Popovici, Samy Bengio, and Jean-Philippe Thiran. "Kernel matching pursuit for large datasets". In: *Pattern Recogn.* 38.12 (2005).

[55] Sam T. Roweis and Lawrence K. Saul. "Nonlinear dimensionality reduction by locally linear embedding". In: *SCIENCE* (2000).

[56] Ameet Talwalkar Sanjiv Kumar Mehryar Mohri. "Sampling methods for the Nystrom method". In: *Journal of Machine Learning Research* (2012).

[57] Andrew M. Saxe et al. "On Random Weights and Unsupervised Feature Learning". In: *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning* (2010).

[58] Walter Scheirer et al. "Robust Fusion: Extreme Value Theory for Recognition Score Normalization". In: *Computer Vision ECCV 2010* 6313 (2010). Ed. by Kostas Daniilidis, Petros Maragos, and NikosEditors Paragios, pp. 481–495.

[59] Bernhard Sch?lkopf et al. "Input Space Versus Feature Space in Kernel-Based Methods". In: *IEEE Transactions on Neural Networks* (1999).

[60] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear component analysis as a kernel eigenvalue problem". In: *Neural Comput.* (1998).

[61] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001. ISBN: 0262194759.

[62] Robert Schreiber, Zeyu Li, and Harlyn Baker. "Robust Software for Computing Camera Motion Parameters". In: *Journal of Mathematical Imaging and Vision* 33.1 (2009).

[63] J. Sivic et al. "Discovering objects and their location in images". In: *ICCV* (2005).

[64] Richard Socher et al. "Convolutional-Recursive Deep Learning for 3D Object Classification". In: *NIPS* (2012).

[65] Michael E. Tipping. "Sparse Kernel Principal Component Analysis". In: *NIPS* (2000).

[66] C. Tomasi. "Bilateral Filtering for Gray and Color Images". In: *ICCV* (1998).

[67] A. Torralba and R. Fergus. "80 Million Tiny Images: a Large Dataset for Non-Parametric Object and Scene Recognition". In: *IEEE PAMI* (2008).

[68] I. Tosic, B.A. Olshausen, and B.J. Culpepper. "Learning Sparse Representations of Depth". In: *Selected Topics in Signal Processing, IEEE Journal of* (2011).

[69] George C Tseng. "Penalized and weighted K-means for clustering with scattered objects and prior information in high-throughput biological data." In: *Bioinformatics* 23.17 (2007), pp. 2247–2255.

[70] Vaibhav Vaish et al. "Using Plane + Parallax for Calibrating Dense Camera Arrays". In: *CVPR* (2004).

[71] S. Wanner and B. Goldluecke. "Globally Consistent Depth Labeling of 4D Lightfields". In: *CVPR*. 2012.

[72] S. Wanner and B. Goldluecke. "Spatial and Angular Variational Super-Resolution of 4D Light Fields". In: *ECCV*. 2012.

[73] Jason Weston et al. "Kernel Dependency Estimation". In: *NIPS*. 2002.

[74] Christopher Williams and Matthias Seeger. "Using the Nystrom Method to Speed Up Kernel Machines". In: *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, pp. 682–688.

[75] John Wright et al. "Robust Face Recognition via Sparse Representation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2009).

[76] Jianchao Yang. "Sparse modeling of high-dimensional data for learning and vision". In: *Ph.D thesis, UIUC* (2011).

[77] Jianchao Yang, Kai Yu, and Thomas S. Huang. "Supervised translation-invariant sparse coding". In: *CVPR* (2010).

[78] Jianchao Yang et al. "Coupled dictionary training for image super-resolution". In: *IEEE Transactions on Image Processing (TIP)* (2012).

[79] Jianchao Yang et al. "Image Super-Resolution Via Sparse Representation". In: *IEEE Trans. on Image Processing* (2011).

[80] Jianchao Yang et al. "Linear spatial pyramid matching using sparse coding for image classification". In: *CVPR* (2009).

[81] Meng Yang and Lei Zhang. "Gabor feature based sparse representation for face recognition with gabor occlusion dictionary". In: *Proceedings of the 11th European conference on Computer vision: Part VI*. ECCV'10. 2010.

[82] Meng Yang et al. "Regularized Robust Coding for Face Recognition". In: *CoRR* (2012).

[83] Jana Kosecka Yi Ma Stefano Soatto and Shankar S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag, 2003.

[84] Kai Zhang and James T Kwok. "Clustered Nystrom method for large scale manifold learning and dimension reduction". In: *IEEE Transactions on Neural Networks* 21.10 (2010), pp. 1576–1587.

[85] Kai Zhang and James T Kwok. "Density-Weighted Nystrom Method for Computing Large Kernel Eigensystems". In: *Neural Computation* 21.1 (2009), pp. 121–146.

[86] Li Zhang et al. "Multiple view image denoising". In: *CVPR*. 2009, pp. 1542–1549.