Learning Hierarchical Sparse Filters for Feature Matching

Jie Fu

a thesis submitted for the degree of

Master of Science

at the University of Otago, Dunedin,

New Zealand.

25 May 2012

Abstract

A common problem in computer vision is to match corresponding points between images. The success of computer vision has usually relied on having good feature representations, which are usually hand-crafted and thus require huge amounts of prior knowledge and human labor. To the best of my knowledge, feature learning algorithms have not been used for feature matching in the literature. In this thesis, I will present how to use feature learning algorithms to build useful feature representations from aerial images in a biologically-inspired and unsupervised manner. These learned feature representations can then be used to do feature matching tasks. Specifically, I will present two algorithms, Sparse Filtering and Deconvolutional Networks for generating hierarchical representations, in which the information from the lower levels is grouped to establish complex features. These complex and hierarchical representations often lead to performance approaching highly hand-designed computer vision algorithms in feature matching tasks.

Acknowledgements

Most of all, I would like to thank my supervisor Brendan McCane. It has been a privilege and an honor to have him as a supervisor. I also would like to thank Areograph Ltd. for giving me the scholarship. I also would like to thank Steven Mills for his invaluable advice and constructive feedback for the research. I also would like to thank my co-supervisor Lubica Benuskova for her invaluable advice for the research and other aspects of academic life. I also would like to thank all the lab members of Artificial Intelligence and Graphics and Vision labs. Finally, I thank my parents.

Contents

1 Introduction **1**
- 1.1 Motivations . . . 1
- 1.2 Contributions . . . 4
- 1.3 Thesis Organization . . . 4

2 Feature Learning **6**
- 2.1 Background: deep learning and sparse coding . . . 7
- 2.2 Patched-based sparse feature learning . . . 8
  - 2.2.1 Previous work . . . 8
  - 2.2.2 Properties of Sparse Filtering . . . 9
  - 2.2.3 Model architecture of Sparse Filtering . . . 10
- 2.3 Hierarchical sparse convolutional feature learning . . . 13
  - 2.3.1 Properties of Deconvolutional Networks . . . 13
  - 2.3.2 Model architecture of Deconvolutional Networks . . . 14

3 Feature Detection **20**
- 3.1 Scale-space extrema detection . . . 20
- 3.2 Key-points localization . . . 23

4 Feature Description and Matching **25**
- 4.1 Forming the descriptors . . . 25
- 4.2 Matching . . . 26
  - 4.2.1 Tentative matches . . . 26
  - 4.2.2 Random Sample Consensus . . . 27

5 Experiments **28**
- 5.1 Settings for SF-based feature matching . . . 28
- 5.2 Settings for DN-based feature matching . . . 29
- 5.3 Buildings images . . . 34
- 5.4 Forest images . . . 35
- 5.5 Varying filter sizes . . . 44
- 5.6 When SIFT fails . . . 49

6 Discussion and Outlook **51**
- 6.1 Relations between the feature detector and descriptor . . . 51
- 6.2 Why only 2 layers? . . . 54

6.3 Speed 57

7 Conclusion **58**

References **59**

A Additional Experiments **66**
A.1 Rocks 66
A.2 Vehicles images 73
A.3 Buildings 79
A.4 Forest images 85
A.4.1 Set 1 85
A.4.2 Set 2 91
A.4.3 Set 3 97
A.4.4 Set 4 103
A.4.5 Set 5 109

List of Tables

2.1 Comparison of tunable parameters in different feature learning algorithms (Ngiam, Koh, Chen, Bhaskar, and Ng, 2011) 10

List of Figures

1.1 Aerial images containing regular shapes (buildings) and aerial images containing irregular shapes (forest) 3
1.2 Overall algorithm flow chart 5

2.1 (Ngiam *et al.*, 2011) The l_2-ball is shown with level sets of the l_1-norm (shown as parallel lines). Left: Two features (e_1,e_2) and two samples (orange and green). After being projected onto the l_2-ball, samples are optimized for sparseness. Right: Two features (blue and green) compete with each other. Because of normalization, the normalized value of feature $\hat{e}_2$ decreases when feature e_1 increases 11
2.2 One layer of Deconvolutional Network 16

3.1 Establishing the DoG scale space (Lowe, 2004) 22
3.2 Detection of maxima and minima of the DoG images (Lowe, 2004) . . . 22

4.1 Build feature histogram using average value pooling upon the 2^{nd} layer latent feature maps (Blum, Springenberg, Wulfing, and Riedmiller, 2011) 26

5.1 A pair of 200×200 flower images as inputs 29
5.2 1^{st} layer and 2^{nd} layer filters learned by SF on flower images 30
5.3 1^{st} layer and 2^{nd} layer filters learned by DN 31
5.4 A pair of forest images 31
5.5 1^{st} layer and 2^{nd} layer filters learned by SF on forest images 32
5.6 Matching reslts of using SF on forest images 33
5.7 Original aerial images of buildings 35
5.8 Extracted patches from aerial images of buildings 36
5.9 1^{st} layer filters on RGB buildings images projected back into pixel space 37
5.10 2^{nd} layer filters on RGB buildings images projected back into pixel space 38
5.11 Original aerial images of forest 40
5.12 Extracted patches from aerial images of forest 41
5.13 1^{st} layer filters on RGB buildings images projected back into pixel space 42
5.14 2^{nd} layer filters on RGB buildings images projected back into pixel space 43
5.15 Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The second pair of forest images. 45

5.16 Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The third pair of forest images. . 46
5.17 Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The forth pair of forest images. . 47
5.18 Average percentage of inliers of varying filter sizes on buildings and forest RGB images using two-layer modles 48
5.19 A pair of image patches extracted from aerial images of forest 49
5.20 6 tentative matches given by SIFT detector and descriptor 49
5.21 2 tentative matches given by SINE with two-layer model 50
5.22 2 tentative matches given by SINE with one-layer model 50

6.1 Matching results after using RANSAC based on SINE on building images 52
6.2 Matching results after using RANSAC based on SINE on RGB forest images 52
6.3 1^{st} layer filters learned by Deconvolutional Networks on buildings and forest 53
6.4 2^{nd} layer and 3^{rd} layer filters learned from two 200×200 images of buildings. Due to the limitation of the RAM (8GB), the number of filters in each layers are 325 and 400 respectively rather than 30 and 465 used in the former experiments with 2 layers models. This figure is best viewed in electronic form. 55
6.5 186 tentative matches using the layer-3 filters on RGB building images 56

A.1 Original aerial images of rocks 67
A.2 Extracted patches from aerial images of rocks 68
A.3 Layer-1 filters on rocks images projected back into pixel space 69
A.4 Layer-2 filters on rocks images projected back into pixel space 70
A.5 Matching results of my proposed method on rocks images 71
A.6 Matching results of SIFT detector and SIFT descriptor on rocks images 72
A.7 Original aerial images of vehicles 73
A.8 Extracted patches from aerial images of vehicles 74
A.9 Layer-1 filters of vehicles projected back into pixel space 75
A.10 layer-2 filters of vehicles projected back into pixel space 76
A.11 Matching results of my proposed method on vehicles images 77
A.12 Matching results of SIFT detector and SIFT descriptor on vehicles images 78
A.13 Original aerial images of buildings (set 2) 79
A.14 Extracted patches from aerial images of buildings (set 2) 80
A.15 Layer-1 filters on buildings images (set 2) projected back into pixel space 81
A.16 Layer-2 filters on buildings images (set 2) projected back into pixel space 82
A.17 Matching results of my proposed method on buildings images 83
A.18 Matching results of SIFT detector and SIFT descriptor on buildings . 84
A.19 Original aerial images of forest (set 1) 85
A.20 Extracted patches from aerial images of forest (set 1) 86
A.21 Layer-1 filters on forest images (set 1) projected back into pixel space . 87

A.22 Layer-2 filters on forest images (set 1) projected back into pixel space . 88
A.23 Matching results of my proposed method on forest images (set 1) . . . 89
A.24 Matching results of SIFT detector and SIFT descriptor on forest images (set 1) . . . 90
A.25 Original aerial images of forest (set 2) . . . 91
A.26 Extracted patches from aerial images of forest (set 2) . . . 92
A.27 Layer-1 filters on forest images (set 2) projected back into pixel space . 93
A.28 Layer-2 filters on forest images (set 2) projected back into pixel space . 94
A.29 Matching results of my proposed method on forest images (set 2) . . . 95
A.30 Matching results of SIFT detector and SIFT descriptor on forest images (set 2) . . . 96
A.31 Original aerial images of forest (set 3) . . . 97
A.32 Extracted patches from aerial images of forest (set 3) . . . 98
A.33 Layer-1 filters on forest images (set 3) projected back into pixel space . 99
A.34 Layer-2 filters on forest images (set 3) projected back into pixel space . 100
A.35 Matching results of my proposed method on forest images (set 3) . . . 101
A.36 Matching results of SIFT detector and SIFT descriptor on forest images (set 3) . . . 102
A.37 Original aerial images of forest (set 4) . . . 103
A.38 Extracted patches from aerial images of forest (set 4) . . . 104
A.39 Layer-1 filters on forest images (set 4) projected back into pixel space . 105
A.40 Layer-2 filters on forest images (set 4) projected back into pixel space . 106
A.41 Matching results of my proposed method on forest images (set 4) . . . 107
A.42 Matching results of SIFT detector and SIFT descriptor on forest images (set 4) . . . 108
A.43 Original aerial images of forest (set 5) . . . 109
A.44 Extracted patches from aerial images of forest (set 5) . . . 110
A.45 Layer-1 filters on forest images (set 5) projected back into pixel space . 111
A.46 Layer-2 filters on forest images (set 5) projected back into pixel space . 112
A.47 Matching results of my proposed method on forest images (set 5) . . . 113
A.48 Matching results of SIFT detector and SIFT descriptor on forest images (set 5) . . . 114

Chapter 1

Introduction

1.1 Motivations

Images are highly variable (because of viewpoint changes, shape variation, etc.) and high-dimensional (often hundreds of thousands of pixels). It is difficult for computer vision algorithms to run on raw image data and to generalize well from training data to unseen data. Therefore, it is desirable to extract features to make computer vision tasks easier to solve (Le, Ngiam, Chen, hao Chia, Koh, and Ng, 2010) and finding "good" feature representations is vital for this kind of high-level computer vision task. Features are defined here as attributes that can be extracted from the input data, and these feature representations could then facilitate subsequent high-level computer vision tasks.

State-of-the-art feature extractors usually consist of a filter bank, a non-linear transformation, and some kind of feature pooling method (Jarrett, Kavukcuoglu, Ranzato, and LeCun, 2009), which collect nearby values through an average, maximum, or histogram computation. The filter bank could be Gabor Wavelets (Serre, Wolf, and Poggio, 2005) or oriented edge detection filters such as Scale Invariant Feature Transform (SIFT) (Lowe, 2004) and Speeded Up Robust Features (SURF) (Bay, Ess, Tuytelaars, and Gool, 2008), and the non-linear transformation can be normalization, winner-take-all, and quantization, etc. For instance, in SIFT which was designed to provide partial invariance to affine and projective transformations (Lowe, 2004), oriented edge filters are applied to a small patch and the dominant orientation is determined by a winner-take-all operation (Juan and Gwun, 2009). After obtaining the resulting sparse vectors, local orientation histograms are produced by adding them over a larger patch (Jarrett *et al.*, 2009).

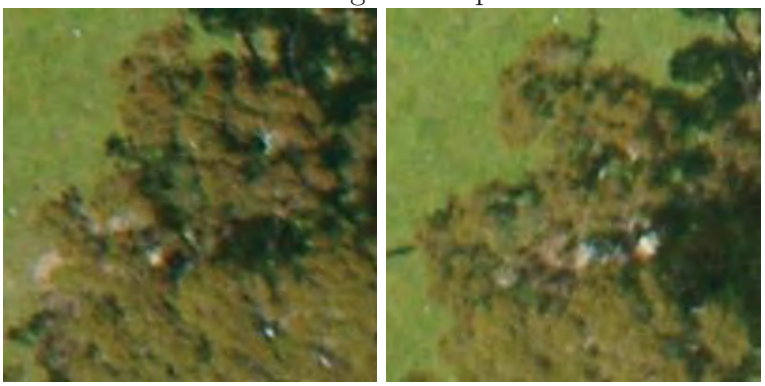
Hand-crafted features, such as SIFT and SURF, are still the most prevailing methods. Other widely used computer vision features include Histograms of Oriented Gradients (HOG) (Dalal and Triggs, 2005), Spin Images (Johnson and Hebert, 1999), Textons (Leung and Malik, 2001) and Shape Context (Leung and Malik, 2001). More recent methods generalize to a broader class of kernel descriptors(Bo, Ren, and Fox, 2010)(Bo, Ren, and Fox, 2011), which highlights the kernel view of orientation histograms and uses a certain type of match filter over image patches (Bo *et al.*, 2010). Typically, these hand-crafted features are low-level concepts (e.g. edges) rather than higher-level concepts such as edge junctions, parallel lines or basic geometric elements, such as rectangles (Zeiler, Krishnan, Taylor, and Fergus, 2010). Although these hand-crafted features are widely used in practice (Lazebnik, Schmid, and Ponce, 2006)(Cheung and Hamarneh, 2007)(Se, Lowe, and Little, 2002)(Juan and Gwun, 2009), there are problems with them.

First, we need to "hand-encode much prior domain-expert knowledge" (Lee, 2010) and to do time consuming hand-tuning (Lee, 2010). Second, multi-stage feature extractions yield better performance than single ones (Jarrett *et al.*, 2009), but there is no prior knowledge that would allow us to design good filters for higher-level abstraction in a hierarchical manner (Jarrett *et al.*, 2009). Admittedly, many hand-crafted detectors, such as Hassir detector (Mikolajczyk and Schmid, 2002), can model mid-level concepts, such as corners. But a hierarchical manner could potentially generate even higher-level concepts, such as fairly complete reconstructions of entire objects (Zeiler, Taylor, and Fergus, 2011).

Much recent research in the machine learning community has focused on learning dictionaries by using sparse coding and deep architectures from training data instead of predefining fixed dictionaries (Mairal, 2010)(LeCun, Kavukcuoglu, and Farabet, 2010)(Ngiam *et al.*, 2011)(Le *et al.*, 2010)(Lee, 2010). Non-smooth convex optimization tools are often applied to the non-convex matrix factorization problem generated by dictionary learning (Mairal, 2010). It has been shown that dictionary learning can be used in computer vision to learn features for modeling the local appearance of natural images, leading to state-of-the-art performance (Mairal, 2010)(LeCun *et al.*, 2010)(Ngiam *et al.*, 2011)(Le *et al.*, 2010)(Lee, 2010). Thus, learning hierarchical feature representations automatically appears to be a more flexible and general strategy (Blum *et al.*, 2011). Some of the learned dictionary elements at the first layer resemble Gabor filters, but they are adapted to the input data, resulting in better results in some cases (Mairal, 2010).



With regular shapes



With irregular shapes

Figure 1.1: Aerial images containing regular shapes (buildings) and aerial images containing irregular shapes (forest)

Nonetheless, it should be noted that a sparse algorithm cannot be stable and vice versa (Xu, Mannor, and Caramanis, 2008), where stability means that given two very similar datasets, an algorithm's output varies little. Therefore, when designing a learning algorithm, one has to make a tradeoff between sparsity and stability, which is often difficult. It was also found in (Rigamonti, Brown, and Lepetit, 2011) that sparsity is not helpful during classification, but it is important when learning the filter bank.

This thesis aims to use flters learned directly from input images to do feature matching and evaluate its effectiveness. In the following chapters, I will show that using these learned filters in a hierarchical biologically-inspired architecture achieves reasonable performance on aerial images containing regular shapes (e.g. buildings1.1) and looks promising on tasks that SIFT is not well suited (e.g. highly irregular forest images 1.1).

1.2 Contributions

Feature learning has been applied mostly to image restoration (Mairal, Bach, Ponce, Sapiro, and Zisserman, 2009) and classification (Ngiam *et al.*, 2011)(Lee, 2010)(Jarrett *et al.*, 2009)(Rigamonti *et al.*, 2011)(Boureau, Bach, LeCun, Ponce, *et al.*, 2010)(Hinton, 2007). To the best of my knowledge, it has not been used for feature point matching in the literature. In this thesis I combine the SIFT detector with multiple stages of trainable robust feature descriptors (based on Deconvolutional Networks) in a purely unsupervised manner for feature point matching tasks. I call this method "SINE", which stands for "SIft and deconvolutional NEtworks". In the following chapters, I will show that SINE yields performance approaching the state-of-the-art in challenging real-world feature point matching tasks on aerial images with a biologically-inspired descriptor.

Since many popular feature descriptors (Lowe, 2004)(Juan and Gwun, 2009)(Leung and Malik, 2001)(Dalal and Triggs, 2005) cannot make use of color information, the ability to easily extract information from color channels is very promising. I will demonstrate that incorporating color information into descriptors is often helpful for the feature point matching problems through a series of experiments.

I discuss the size of filters used to infer feature descriptors, and experimental results show that larger sizes are preferable under certain circumstances.

1.3 Thesis Organization

The flow chart of the overall algorithm is shown in Fig. 1.2. Firstly, Chapter 2 will present how to use Sparse Filtering and Deconvolutional Networks to produce feature maps from two input images. Next, how to detect interest points of each input image will be presented in Chapter 3. Chapter 4 will first present how to form a pooling-based descriptor by grouping the region around those interest points, and then demonstrate how to use a nearest neighbor search to detect the similarities between feature descriptors of the two images. Chapter 5 will present a series of experiments. Chapter 6 will provide a discussion and outlook. Chapter 7 will give the final conclusion. Additional experiments using two-layer convolution-based models will be given in the Appendix.

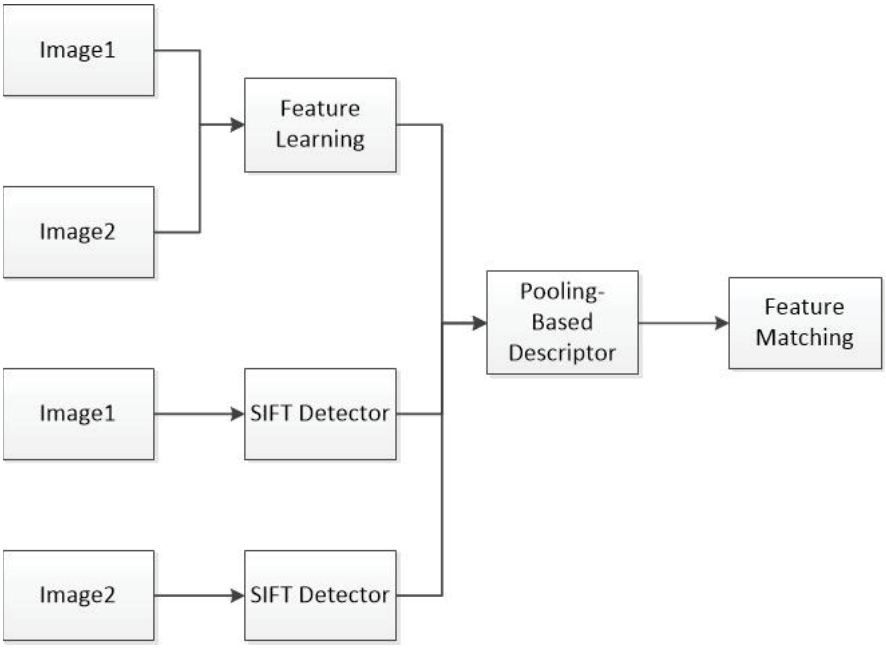


Figure 1.2: Overall algorithm flow chart

Chapter 2

Feature Learning

It appears that the visual system decomposes objects into different features, e.g. edges, colors and shapes (Wallisch, 2009). In (Olshausen and Field, 1997), it is suggested that V1, the first layer of the visual cortex, constructs a sparse representation of the images. It is also assumed in the literature of unsupervised feature learning (Chen, Polatkan, Sapiro, Dunson, and Carin, 2011)(Coates, Lee, and Ng, 2010)(Boureau *et al.*, 2010)(Blum *et al.*, 2011)(Ranzato, Huang, Boureau, and LeCun, 2007)(Le *et al.*, 2010) that structures in unlabeled data are useful for subsequent machine learning and computer vision tasks. To be specific, for feature point matching, if we can discover higher-level structures like parts of an object appearing in both images, it becomes easier to match the corresponding points. State-of-the-art hierarchical unsupervised feature learning methods can extract both low-level structures such as edges and higher-level structures such as edge junctions and elemental shapes (Boureau *et al.*, 2010). There are two kinds of feature learning methods: the patch-based one is trained from randomly selected patches in the original images, which produce a highly redundant representation for the whole image (Rigamonti *et al.*, 2011)(Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu, and LeCun, 2010); the convolution-based method is trained over the whole image at once, which effectively reduces the redundancy (Kavukcuoglu *et al.*, 2010). Although in this thesis I focus on a convolution-based model, Deconvolutional Networks (Zeiler *et al.*, 2010), I also evaluate Sparse Filtering (Ngiam *et al.*, 2011) on a pair of grayscale images, which is a patch-based feature learning algorithms.

2.1 Background: deep learning and sparse coding

Both Sparse Filtering (SF) and Deconvolutional Networks (DN) are based on sparse coding (Olshausen and Field, 1997)(Mairal, 2010) and a deep learning architecture (Bengio, 2009)(Hinton and Salakhutdinov, 2006)(Hinton, 2007). More recent theoretical results (Bengio, 2009) suggest that to learn complicated functions that can represent high-level abstractions (e.g. in computer vision), deep learning architecture may be needed. For instance, if a solution to the computer vision task is "represented with a very large but shallow architecture (with many computational elements)" (Bengio, 2009), an exponential number of training images might be needed to tune each of these elements and capture a highly-varying function (Bengio, 2009). Clearly, anything that is exponential is of no use for real-world tasks and poor generalization may also be expected. Deep learning architectures "learn complex mappings by transforming their inputs through multiple layers of nonlinear operations" (Hinton, Osindero, and Teh, 2006).

The information from the lower levels is grouped to establish complex features. By imposing some sparse norm for the regularization at every level of the hierarchical model, feature patterns are naturally becoming more complex (Kavukcuoglu *et al.*, 2010).

Sparse coding was first introduced by the computational neuroscience community with ICA, but it has been extensively used in the computer vision community (Kanan and Cottrell, 2010)(Mairal, 2010)(Coates and Ng, 2011). Sparse coding is inspired by the phenomenon that inhibitory neurons enable the information in the brain to be represented by a small percentage of active neurons within a large population of neurons (Hawkins and George, 2006). A key property of sparse coding representations is that we only need to match a portion of the pattern to guarantee the matching is significant (Gao, Tsang, Chia, and Zhao, 2010)(Yang, Yu, Gong, and Huang, 2009). Sparse coding is the approximation of signals, which can also be viewed as an extension of the classical wavelet framework (Mairal, 2010). These signals are linear combinations of the minimum number of elements from a dictionary (Bouvrie, 2009). Sparse coding does not require that the basis vectors be orthogonal, resulting in more flexibility to adapt to the input data, which is very different from those decompositions based on PCA and its variants (Elad and Aharon, 2006).

2.2 Patched-based sparse feature learning

2.2.1 Previous work

Features can be extracted from unlabeled data by many unsupervised learning algorithms, and I will briefly discuss some of them.

K-means clustering has been widely used in computer vision (Coates *et al.*, 2010). It is adaptive and has few parameters to tune. K-means aims to partition a set of training examples into k clusters in which each training example is assigned to the cluster with the nearest mean. It was shown that (Coates *et al.*, 2010) when applied to whitened (Rigamonti *et al.*, 2011) image data (called sphering, the goal is to make the input less redundant), K-means can learn meaningful feature representations such that it has demonstrated good performance on the classification dataset CIFAR-10 (Krizhevsky and Hinton, 2009). The main drawback of K-means is that it can only form a single layer of feature representation thus making it impossible to capture higher-level structures.

Principal Component Analysis (PCA) (Stone, 2004) is a popular subspace learning (Wright, Ma, Mairal, Sapiro, Huang, and Yan, 2010) algorithm for preprocessing and dimensionality reduction. Other similar algorithms include Independent Subspace Learning (ISA) (Le, Karpenko, Ngiam, and Ng, 2011) which is a variant of Independent Component Analysis (ICA) and Linear Discriminant Analysis (LDA) (Belhumeur, Hespanha, and Kriegman, 1997), and they can all be explained within the graph embedding framework (Wright *et al.*, 2010). The key idea of PCA is to extract a set of basis vectors from unlabeled data while retaining as much as possible of the "variability" of the data (Jolliffe and MyiLibrary, 2002) and the original data can be approximately recovered by the projected coordinates (Lee, 2010). Because PCA has the implicit assumption that the features observe a Gaussian distribution (Zhou, Zhuang, Tang, Hasegawa-Johnson, and Huang, 2008), when dealing with data with a highly non-Gaussian distribution, it will not work very well.

In fact, PCA can also be categorized into a type of matrix factorization (Lee, 2010). Given the input data $X \in \mathbb{R}^{d \times m}$(where each column of X represents an individual observation, d is the dimension of the input data, and m is the number of observations), matrix factorization algorithms decompose it in the form of $X \approx B\,S$ (where $B \in \mathbb{R}^{d \times n}$ and $S \in \mathbb{R}^{n \times m}$, n is the number of basis vectors, and each column of B represents an individual basis vector). In other words, input data can be decomposed as a linear combination of basis vectors by matrix factorization algorithms (Stone, 2004), and

each column of S acts as the coefficients of the basis vectors for each input example (Lee, 2010).

Independent Component Analysis (ICA) (Hyvärinen, Karhunen, and Oja, 2001) is widely used for blind source separation. It is also related to PCA and Factor Analysis (FA) (Stone, 2004). PCA and FA attempt to find a set of basis vectors which are *uncorrelated* with each other, whereas ICA finds a set of *independent* basis vectors instead (Stone, 2004). Informally speaking, independence implies a lack of correlation, but a lack of correlation does not imply independence (Lee, 2010). Therefore, the property of basis vectors discovered by PCA and FA is much weaker than that of ICA's (Stone, 2004). It was shown that (Le *et al.*, 2010) when a modified Topographic ICA (Hyvärinen *et al.*, 2001) is combined with Convolutional Neural Networks (CNN) (LeCun *et al.*, 2010), it can learn complex invariant features. However, ICA can only produce a linear function of the input data which makes it unsuitable for noisy datasets (Lee, 2010).

2.2.2 Properties of Sparse Filtering

Many feature learning algorithms construct the explicit models that provide good approximations of the data distribution. These include ICA , Sparse Coding (Mairal, 2010), and Sparse Restricted Boltzmann Machines (RBMs) (Hinton *et al.*, 2006). By contrast, Sparse Filtering (SF) (Ngiam *et al.*, 2011) only tries to optimize for sparsity in the *feature distribution* rather than modeling the *data distribution* explicitly, thus having a simple formulation and enabling efficient and large-scale learning.

Feature learning algorithms can be seen as generating particular feature distributions (Ngiam *et al.*, 2011). Suppose there is a feature distribution matrix over a dataset, where each row represents a feature, each column represents a sample patch, and each element $e_j^{(i)}$ is the response of feature j on sample patch i. SF works on searching for the following properties of the feature distribution (Ngiam *et al.*, 2011):

- Sparse feature per sample (Population sparsity). For each column $e^{(i)}$ (sample patch) in the feature matrix, there should be a small number of non-zero values. It is believed that (Willmore and Tolhurst, 2001)(Field, 1994) the early visual cortex adopts this property as its key principle of coding.

- Sparse features across samples (Lifetime sparsity (Willmore and Tolhurst, 2001)). In order to keep features discriminative, each row in the feature matrix should have few non-zero values.

Algorithm	Tunable hyper-parameters
Sparse Filtering	# features
ICA	# features
Sparse Coding	# features, sparsity penalty, mini-batch size
Sparse RBMs	# features, target activation, weight decay, sparsity penalty, momentum

Table 2.1: Comparison of tunable parameters in different feature learning algorithms (Ngiam *et al.*, 2011)

- Uniform activity distribution (High dispersal). Each row should have similar activities. This is achieved by averaging the squared values in the feature matrix across the columns (samples). Though this property is not absolutely necessary for good feature representations, it would prevent the same features being active at the same time (Ngiam *et al.*, 2011).

In (Ngiam *et al.*, 2011), it is found that the properties of population sparsity and high dispersal are jointly sufficient to ensure lifetime sparsity. Due to its simple formulation, SF has very few hyper-parameters to tune[1] as shown in Tab. 2.1. Because different settings for those hyper-parameters lead to different performances (Ngiam *et al.*, 2011), essentially hyper-parameter-free SF can fit a large amount of datasets easily by optimizing the sparsity of l_2-normalized features only.

2.2.3 Model architecture of Sparse Filtering

Let us use $e_j^{(i)}$ to represent the j^{th} feature value (row) for the i^{th}sample patch (column):

$$e_j^{(i)} = w_j^T x^{(i)}, \tag{2.1}$$

where w_j represents the j^{th} feature (coefficients, which can be seen as *filters* to be learned), and $x^{(i)}$ represents the i^{th} sample.

SF first normalizes the feature distribution matrix by rows. In other words, SF divides each feature by its l_2-norm over all samples:

$$\tilde{e}_j = \frac{e_j}{\| e_j \|_2}. \tag{2.2}$$

Then SF normalizes these features per sample by calculating:

[1]When learning over-complete feature representations, ICA needs very expensive approximate orthogonalization algorithms (Hyvrinen, Hurri, and Hoyer, 2009)

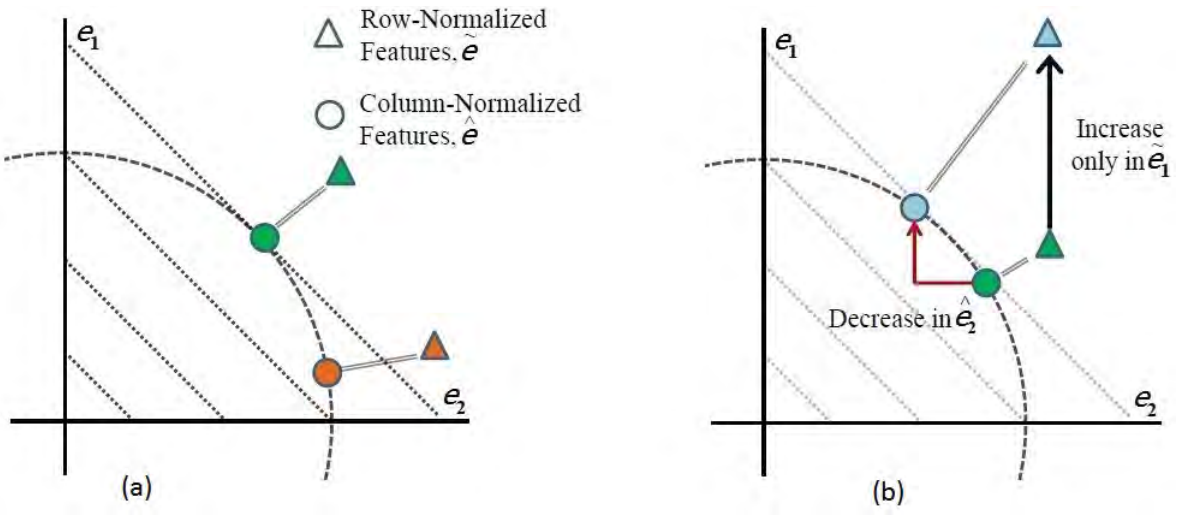


Figure 2.1: (Ngiam *et al.*, 2011) The l_2-ball is shown with level sets of the l_1-norm (shown as parallel lines). Left: Two features (e_1,e_2) and two samples (orange and green). After being projected onto the l_2-ball, samples are optimized for sparseness. Right: Two features (blue and green) compete with each other. Because of normalization, the normalized value of feature $\hat{e}_2$ decreases when feature e_1 increases .

$$\hat{e}^{(i)} = \frac{\tilde{e}^{(i)}}{\| \tilde{e}^{(i)} \|_2}. \tag{2.3}$$

Finally, these normalized features are optimized by introducing the l_1 penalty. When dealing with M sample image patches, SF tries to minimize the following objective:

$$\sum_{i=1}^{M} \| \hat{e}^{(i)} \| = \sum_{i=1}^{M} \left\| \frac{\tilde{e}^{(i)}}{\| \tilde{e}^{(i)} \|_2} \right\|_1 . \tag{2.4}$$

Optimize for population sparsity

The population sparsity of the features on the i^{th} sample is measured by $\| \hat{e}^{(i)} \|_1 = \left\| \frac{\tilde{e}^{(i)}}{\|\tilde{e}^{(i)}\|_2} \right\|_1$, and can be minimized when the features are sparse as is shown in Fig. 2.1-(a). All the other $\hat{e}_j^{(i)}$ will decrease even if only one $e^{(i)}$ increases as shown in Fig. 2.1-(b). Therefore some features in $e^{(i)}$ become very large while most of them are close to zero.

It should be noted that Eq. 2.4 is related to the Treves-Rolls (Ngiam *et al.*, 2011)(Willmore and Tolhurst, 2001) measure of lifetime sparsity:

$$s^{(i)} = \frac{\left[\sum_j \frac{\tilde{e}_j^{(i)}}{E}\right]^2}{\left[\sum_j \frac{(\tilde{e}_j^{(i)})^2}{E}\right]}, \tag{2.5}$$

where E is the number of features. SF can be seen as a re-scale of the square-root of the formulation above, which is usually used to model the sparsity of neuron activations (Ngiam *et al.*, 2011).

Optimize for high dispersal

High dispersal can be achieved by making the mean squared activation of each feature to be approximately equal, each feature has to be normalized first by being divided by its norm:

$$\tilde{e}_j = \frac{e_j}{\| e_j \|_2}. \tag{2.6}$$

Optimize for lifetime sparsity

As discussed above, population sparsity ensures that a feature distribution have many zero elements. Furthermore, these features are highly dispersed, which means that those zero elements are evenly distributed across all the features. Therefore, optimizing for population sparsity and high dispersal are sufficient to ensure lifetime sparsity (Ngiam *et al.*, 2011).

Deep architecture

In (Ngiam *et al.*, 2011), a deep training architecture is built by using a greedy layer-wise approach (Bengio, Lamblin, Popovici, and Larochelle, 2007). Specifically, after the 1^{st} layer of features are trained, the normalized features $\hat{e}^{(i)}$ will be used as input for learning the 2^{nd} layer of features.

However, through experiments, I found that[2] the 2^{nd} layer filters are very similar to the 1^{st} layer filters, except for different sizes (will be demonstrated in Sec. 5.1). Thus when doing feature matching experiments, I will only use a one-layer SF model.

[2]Similar discoveries were also found in (Zeiler *et al.*, 2010)

2.3 Hierarchical sparse convolutional feature learning

Most state-of-the-art feature learning algorithms are trained at the patch level (Ngiam *et al.*, 2011)(Lee, 2010)(Ranzato *et al.*, 2007)(Jarrett *et al.*, 2009)(Rigamonti *et al.*, 2011). After filters are learned, they extract feature representations by convolving images with learned filters (Jarrett *et al.*, 2009), resulting in highly redundant representations due to the fact that overlapping patches are encoded in isolation(Kavukcuoglu *et al.*, 2010). Patch-based feature learning only produces oriented edge detectors, but convolution-based feature learning can produce highly diverse filters, e.g. corner detectors (Kavukcuoglu *et al.*, 2010). In the following chapters, I will use the Deconvolutional Network (DN) (Zeiler *et al.*, 2010), which is a convolutional feature learning algorithm proposed by Matthew Zeiler, to form the hierarchical sparse convolutional feature representations in a purely unsupervised fashion. Convolutional training reduces the redundancy of the feature representations(Zeiler *et al.*, 2010).

2.3.1 Properties of Deconvolutional Networks

Although a sparse representation is not necessarily crucial for feature extraction during matching, it is important for learning the filters (Rigamonti *et al.*, 2011). Moreover, sparsity alone is not enough and a effective and efficient architecture is needed (Zeiler *et al.*, 2010). The following lists the key properties of DN:

1. The locality can be preserved by using stable latent representations introduced by a convolutional approach (LeCun *et al.*, 2010)(Kavukcuoglu *et al.*, 2010)(Zeiler *et al.*, 2010). With the convolutional approach, DN can extract meaningful mid-level features (e.g. corners, edge junctions, parallel lines, and etc.). By contrast, most deep feature learning algorithms are not based on convolution (LeCun *et al.*, 2010)(Le *et al.*, 2011)(Olshausen and Field, 1997)(Ngiam *et al.*, 2011)(Wright *et al.*, 2010).

2. DN is different from the Convolutional Networks (CN) proposed by (LeCun *et al.*, 2010) in that CN is a bottom-up approach, in which the input is processed through a series of convolutions, non-linearities and sub-sampling (Zeiler *et al.*, 2010)(Kavukcuoglu *et al.*, 2010). But DN uses a top-down method, which enables it to produce the input by summing over convolutions of the latent feature maps with learned filters (Zeiler *et al.*, 2010).

3. Like Deep Belief Networks (Lee, Grosse, Ranganath, and Ng, 2009) and other hierarchical unsupervised sparse auto-encoders (Jarrett *et al.*, 2009), DN greedily forms layers from the input images upwards. The distinction is that there is no encoder in DN whereas other methods have both the encoder and the decoder(Zeiler *et al.*, 2010). The encoder maps input into the latent feature space in a bottom-up fashion, and the decoder gives a mapping from the feature space back to the input space, trying to approximate the original input images. The encoders are usually simple non-linear functions, which largely restrict the feasible latent representations, leading to sub-optimal features (Zeiler *et al.*, 2010). DN works without the encoder, hopefully learning better features by producing the features exactly (rather than approximating by using an encoder) (Zeiler *et al.*, 2010).

2.3.2 Model architecture of Deconvolutional Networks

Pre-processing

It is crucial to remove the linear dependencies between the coefficients of the input images for feature learning algorithms to converge (Rigamonti *et al.*, 2011). Many feature learning algorithms whiten input images first (Rigamonti *et al.*, 2011)(Zeiler *et al.*, 2010)(Ngiam *et al.*, 2011). A whitening pre-procesing matrix W can be formed by:

$$W = ED^{-1/2}E^{\top}, \tag{2.7}$$

where E can be computed from $C = EDE^{\top}$ and C is the covariance matrix of input images. However the whitening operation is not workable for reasonable large images due to the need to compute the covariance matrix of the input images (Rigamonti *et al.*, 2011).

Furthermore, the human visual system is believed to utilize local contrast normalization to get the desired dynamic range (Behnke, 2003)(Boynton, Demb, Glover, and Heeger, 1999). It is also shown in (Jarrett *et al.*, 2009) that local contrast normalization is one of the most important components for feature learning results. While many feature learning algorithms (Jarrett *et al.*, 2009) require an expensive and impractical cubic-time whitening step, DN employs a local contrast normalization (Brady and Field, 2000) pre-processing step.

The local contrast normalization is inspired by computational neuroscience models, which consists of local subtractive and divisive normalizations (Jarrett *et al.*, 2009)(Zeiler *et al.*, 2010). For a given location x_{ij}, the subtractive normalization can be computated as:

$$\alpha_{ij} = x_{ij} - \sum_{pq} w_{pq} \cdot x_{i+p,j+q}, \tag{2.8}$$

where w_{pq} is a Gaussian filter (of size 13×13 in this thesis) such that $\sum_{pq} w_{pq} = 1$.

The divisive normalization can be done with:

$$\beta_{ij} = \frac{\alpha_{ij}}{max(c, \sigma_{ij})}, \tag{2.9}$$

where $\sigma_{ij} = (\sum_{pq} w_{pq} \cdot \alpha^2_{i+p,j+q})^{1/2}$, and the constant $c = mean(\sigma_{ij})$.

A single layer

Suppose that the ith image y^i, which consists of K_0 color channels $y^i_1, ..., y^i_{K_0}$, is input to this single layer. The K_1 latent feature maps z^i_k are first convolved with learned filters $f_{k,c}$ and then summed linearly to form each color channel c of the input image y^i:

$$\sum_{k=1}^{K_1} z^i_k \oplus f_{k,c} = y^i_c, \tag{2.10}$$

where $\oplus$ is the convolution operation. It is easy to see that if y^i_c is an $N_r \times N_c$ image and the learned filters are $H \times H$, then the size of the latent feature maps z is $(N_r + H - 1) \times (N_c + H - 1)$. It should be noted that 2.10 is an ill-posed equation, such that we have to impose an extra regularization term on z^i_k to make the latent feature maps sparse. By introducing the sparsity, we can obtain a unique solution. Now we have an overall cost function:

$$C_l(y^i) = \frac{\lambda}{2} \sum_{c=1}^{K_0} \| \sum_{k=1}^{K_1} z^i_k \oplus f_{k,c} - y^i_c \|^2_2 + \sum_{k=1}^{K_1} |z^i_k|^p, \tag{2.11}$$

where we assume some sparse norm p for the regularization and Gaussian noise. In fact, $|w|^p = \sum_{i,j} |w(i,j)|^p$ and λ is a constant value used to balance the sparsity of the latent feature maps z^i_k and the weighted contributions of the reconstruction of the image y^i.

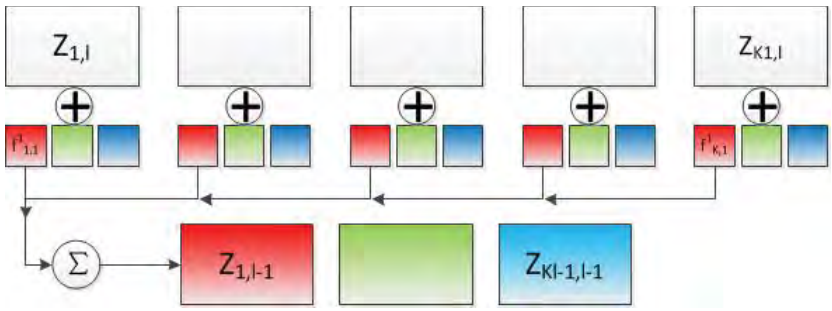


Figure 2.2: One layer of Deconvolutional Network

During the learning process, we use a pair of images $y = \{y^1, y^2\}$ to solve $argmin_{f,z} C_l(y)^2$, in which the filters are the same for the two images but the images have different latent feature maps. These latent feature maps then form the descriptors.

Forming a deep learning architecture

A hierarchical model can be easily formed: the latent feature maps $z^i_{k,l}$ of layer l can be treated as the input for layer $l+1$. Therefore the cost function C_l for layer l is:

$$C_l(y) = \frac{\lambda}{2} \sum_{i=1}^{I} \sum_{c=1}^{K_{l-1}} \| \sum_{k=1}^{K_l} g^l_{k,c}(z^i_{k,l} \oplus f^l_{k,c}) - z^i_{c,l-1} \|_2^2 + \sum_{i=1}^{I} \sum_{k=1}^{K_l} |z^i_{k,l}|^p, \tag{2.12}$$

where $g^l_{k,c}$ represents elements of an indexing binary matrix that determines the connectivity between the adjacent latent feature maps and $z^i_{c,l-1}$are the latent feature maps from the lower layer. Although it is assumed that $g^1_{k,c}$ in the 1^{st} layer is 1, $g^l_{k,c}$ in higher layers should be sparse. As is shown in Fig. 2.2 the hierarchical model is trained in a bottom-up fashion.

Learning filters

$C_l(y)$ has to be minimized with respect to the latent feature maps while keeping the filters unchanged and then we minimize $C_l(y)$ with respect to the filters while keeping the latent feature maps unchanged. The series of optimization operations are done in a layer-wise way, which means taking the images y as the input to the 1^{st} layer.

Inferring optimal latent feature maps In order to learn non-trivial solutions, DN imposes a sparsity constraint on the latent feature maps $z^i_{k,l}$. When we set $p = 1$, the optimization problem for the latent feature maps can be resolved by certain techniques (Lee, Battle, Raina, and Ng, 2007). However, the problem is poorly conditioned

because every element in the latent feature maps is coupled to one another via the filters (Zeiler *et al.*, 2010), which means that it may take a long time to converge to an acceptable solution.

Therefore, DN adopts a type of continuation method (Wang, Yang, Yin, and Zhang, 2008)(Geman and Yang, 1995) to construct a general framework which is suitable for any value of $p > 0$. DN minimizes an auxiliary cost function $\hat{C}_l(y)$ which introduces extra variables $x^i_{k,l}$ with respect to each value in the latent feature maps $z^i_{k,l}$ rather than minimizing Eq. 2.12 directly:

$$\hat{C}_l(y) = \frac{\lambda}{2}\sum_{i=1}^{I}\sum_{c=1}^{K_{l-1}} \| \sum_{k=1}^{K_l} g^l_{k,c}(z^i_{k,l} \oplus f^l_{k,c}) - z^i_{c,l-1}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{I}\sum_{k=1}^{K_l} \|z^i_{k,l} - x^i_{k,l}\|_2^2 + \sum_{i=1}^{I}\sum_{k=1}^{K_l} |x^i_{k,l}|^p, \tag{2.13}$$

where β is a continuation parameter. By incorporating the auxiliary variables $x^i_{k,l}$ to separate the convolution part of the cost function from the $|\cdot|^p$ term, we can utilize an alternating form of minimization for $z^i_{k,l}$. A quadratic problem for $z^i_{k,l}$ is formed by first fixing $x^i_{k,l}$, which is called z sub-problem. $z^i_{k,l}$ is then fixed and a separable 1D problem for each element in auxiliary variables $x^i_{k,l}$ is solved, which is called x sub-problem. As DN switches between these two alternating steps, β is increased from a small value until it forces $z^i_{k,l}$ to get close to $x^i_{k,l}$.

z sub-problem It can be seen from Eq. 2.13 that $z^i_{k,l}$ can be solved independently. Assuming a fixed $x^i_{k,l}$, the derivatives of $\hat{C}_l(y)$ are computed with respect to $z^i_{k,l}$:

$$\frac{\partial \hat{C}_l(y)}{\partial z^i_{k,l}} = \lambda \sum_{c=1}^{K_{l-1}} F^{l^T}_{k,c}(\sum_{\tilde{k}=1}^{K} F^l_{\tilde{k},c} z^i_{k,l} - z^i_{c,l-1}) + \beta(z^i_{k,l} - x^i_{k,l}), \tag{2.14}$$

where when $g^l_{k,c} = 1$, $F^l_{k,c}$ is a sparse convolution matrix which is equivalent to convolving with $f^l_{k,c}$; when $g^l_{k,c} = 0$, $F^l_{k,c} = 0$. Since different latent feature maps compete with each other to explain local patterns in the most compact fashion, all the latent feature maps $z^i_{k,l}$ must be optimized simultaneously for a fixed i and varying k. When $\frac{\partial \hat{C(y)}}{\partial z^i_{k,l}} = 0 \; \forall k$, given a fixed i, the following $K_l(N_r + H - 1)(N_c + H - 1)$linear system gives the solution to the optimal $z^i_{k,l}$:

$$A \begin{pmatrix} z^i_{1,l} \\ \cdot \\ z^i_{K_l,l} \end{pmatrix} = \begin{pmatrix} \sum_{c=1}^{K_{l-1}} F^{l^T}_{1,c} z^i_{c,l-1} + \frac{\beta}{\lambda} x^i_{1,l} \\ \cdot \\ \sum_{c=1}^{K_{l-1}} F^{l^T}_{K,c} z^i_{c,l-1} + \frac{\beta}{\lambda} x^i_{K_l,l} \end{pmatrix}, \tag{2.15}$$

where

$$A = \begin{pmatrix} \sum_{c=1}^{K_{l-1}} F_{1,c}^{l^T} F_{1,c}^{l} + \frac{\beta}{\lambda} I & \cdot & \sum_{c=1}^{K_{l-1}} F_{1,c}^{l} F_{K_l,c}^{l^T} \\ \cdot & \cdot & \cdot \\ \sum_{c=1}^{K_{l-1}} F_{K_l,c}^{l} F_{1,c}^{l^T} & \cdot & \sum_{c=1}^{K_{l-1}} F_{K_l,c}^{l^T} F_{K_l,c}^{l} + \frac{\beta}{\lambda} I \end{pmatrix}. \tag{2.16}$$

We can use conjugate gradient descent to minimize Eq. 2.15 efficiently. Because of the filter size I use in this thesis is 19×19 pixels (nearly 3 times larger than other papers), the convolution problems are solved in the spatial domain rather than using FFTs, which would introduce serious boundary effects in the latent feature maps (Zeiler *et al.*, 2010).

x **sub-problem** When $z_{k,l}^i$ is fixed, we need to solve a 1D optimization problem for each value of the latent feature maps to find the best $x_{k,l}^i$. Specifically if $p = 1$, the closed-form solution of $x_{k,l}^i$ can be give by (Wang *et al.*, 2008):

$$x_{k,l}^i = max(|z_{k,l}^i| - \frac{1}{\beta}, 0) \frac{z_{k,l}^i}{|z_{k,l}^i|}, \tag{2.17}$$

where all the computations are done in an element-wise fashion. For arbitrary values of $p > 0$, a lookup-table (Krishnan and Fergus, 2009) can be used to give the optimal solution.

Filters update Given fixed x and computed $z_{k,l}^i$ for a fixed i, filter $f_{k,c}^l$ can be updated:

$$\frac{\partial \hat{C_l(y)}}{\partial f_{k,c}^l} = \lambda \sum_{i=1}^{I} \sum_{\tilde{c}=1}^{K_{l-1}} Z_{k,l}^{i^T} (\sum_{\tilde{k}=1}^{K_l} g_{\tilde{k},c}^l Z_{\tilde{k},l}^i f_{\tilde{k},\tilde{c}}^l - z_{\tilde{c},l-1}^i), \tag{2.18}$$

where Z is a convolution matrix similar to F.

The overall learning algorithm is as follows (Zeiler *et al.*, 2010):

- Input: Training images y
- Parameters: Number of latent feature maps K, connectivity g, regularization parameter λ, number of epochs E, continuation parameters: $\beta_0, \beta_{Inc}, \beta_{Max}$

1. Initialize latent feature maps and filters randomly $z \sim N(0, \epsilon), f \sim N(0, \epsilon)$
2. for epoch $= 1 : E$ do

3. for $i = 1 : I$ do

4. $\beta = \beta_0$

5. while $\beta < \beta_{Max}$ do

6. Given $z^i_{k,l}$, solve for $x^i_{k,l}$ based on Eq. 2.17, $\forall k$

7. Given $x^i_{k,l}$, solve for $z^i_{k,l}$ based on Eq. 2.15, $\forall k$

8. $\beta = \beta \cdot \beta_{Inc}$

9. end while

10. end for

11. Update $f^l_{k,c}$ based on gradient descent Eq. 2.18, $\forall k, c$

12. end for

13. Output: filters f

Forming latent feature maps

The model I use in this thesis consists of 2 layers. First I project the 2^{nd} layer latent feature maps $z_{k,2}$ through $f^2_{k,c}$ and $f^1_{k,c}$ onto the image in order to minimize the reconstruction error with respect to y':

$$\frac{\lambda_2}{2}\sum_{c=1}^{K_0}\|\sum_{k=1}^{K_1} g^1_{k,c}(\sum_{b=1}^{K_2} g^2_{b,k}(z_{b,2} \oplus f^2_{b,k})) \oplus f^1_{k,c}) - y'_c\|_2^2 + \sum_{k=1}^{K_2}|z_{k,2}|. \tag{2.19}$$

Then I minimize the error with respect to $z_{k,2}$:

$$\frac{\lambda_2}{2}\sum_{c=1}^{K_1}\|\sum_{k=1}^{K_2} g^2_{k,c}(z_{k,2} \oplus f^2_{k,c}) - z_{c,1}\|_2^2 + \sum_{k=1}^{K_2}|z_{k,2}|. \tag{2.20}$$

By using Eq. 2.19 and Eq. 2.20 alternatively, the 2^{nd} layer latent feature maps $z_{k,2}$ will converge (Zeiler *et al.*, 2010).

When $z_{k,2}$ has converged, y' can also be reconstructed by projecting back to the image through $f^2_{k,c}$ and $f^1_{k,c}$:

$$\tilde{y'} = \sum_{k=1}^{K_1} g^1_{k,c}(\sum_{b=1}^{K_2} g^2_{b,k}(z_{b,2} \oplus f^2_{b,k})) \oplus f^1_{k,c}. \tag{2.21}$$

Chapter 3

Feature Detection

Feature detection is the process of deciding where to sample an image, resulting in a set of key-points that specify locations in the image. There are three different sub-fields of detection: the first group focuses on finding key-points that are stable under minor affine and photometric transformations (O'Hara and Draper, 2011), which is often referred to as an Interest Point Operator(Lowe, 2004); the second group relies on mathematical models of the human visual attention system (Kanan and Cottrell, 2010); some other research samples the images by using a multiscale framework, or even by random sampling (O'Hara and Draper, 2011).

This thesis focuses on showing that using a trainable feature descriptor can improve the performance in feature point matching tasks, so the standard SIFT detector is used in this thesis in order to compare my proposed method with the SIFT method (including the detector and the descriptor). The detector used by the standard SIFT method is an Interest Point Operators and uses a multiscale framework to search for blobs in images (Lowe, 2004)(Lowe, 1999). By using a cascade of difference-of-Gaussian filters, the SIFT framework searches for robust features at different scales to detect locations that are invariant to scale changes of the image(Lowe, 2004). The following describes how SIFT locates the key-points, which will then be used to guide feature learning and feature matching.

3.1 Scale-space extrema detection

By constructing a Gaussian pyramid and detecting the extrema (both maxima and minima) in a series of Difference-of-Gaussian (DoG) scale space which are invariant to scale and orientation, SIFT can determine the locations and scales of potential interest

points in images (Lowe, 2004). It should be noted that my proposed feature matching algorithm (SINE) does not depend on the scale information. Gaussian scale-space (GSS) is firstly needed to construct DoG scale space. Given an input image, $I(x, y)$, the GSS of an image is defined as a function, $L(x, y, \sigma)$, which is computed by the convolution of a variable-scale Gaussian, $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) \oplus I(x, y), \tag{3.1}$$

where $\oplus$ is the convolution operation in x and y, and $G(x, y, \sigma)$ is a 2-dimensional Gaussian distribution function:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{3.2}$$

The GSS is made of a series of smoothed images with different σ over a number of octaves in which the size of the image is down-sampled by 2 at each octave. Due to the fact that $L(x, y, \sigma)$ are blurred with increasing values of σ, images of the next octave can be re-sampled without losing too much information. The σ of GSS is set in logarithmic steps divided into O octaves, in which each octave is subdivided into S scales. Given a value of octave o and scale level s, the value of $\sigma$is given by

$$\sigma(o, s) = \sigma_o \cdot 2^{(o+\frac{s}{S})}, \tag{3.3}$$

where $s \in [0, S-1], o \in [0, O-1]$ and σ_0 is the base scale level.

An efficient way of constructing $D(x, y, \sigma)$ is shown in Fig. 3.1. The input image is convolved with Gaussian filters at different scales, which is shown stacked in the left column of Fig. 3.1. Each octave of scale space, which is doubling the value of σ, is divided into an integer number, s, of intervals, meaning that $k = 2^{1/s}$. On the right column Fig. 3.1, the difference-of-Gaussian images are generated by subtracting the adjacent image scales.

In order to detect stable key-point locations in scale space, a variable scale Gaussian, whose result of convolving an image with a DoG function is given by

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \star I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma), \tag{3.4}$$

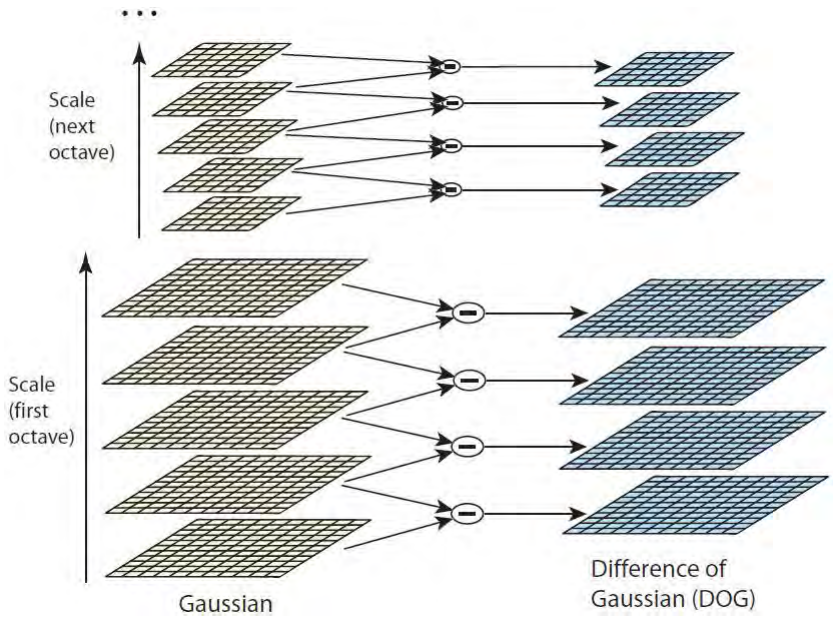


Figure 3.1: Establishing the DoG scale space (Lowe, 2004)

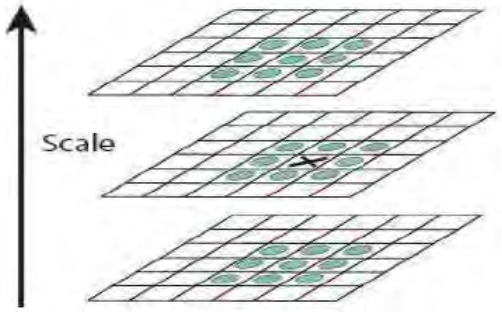


Figure 3.2: Detection of maxima and minima of the DoG images (Lowe, 2004)

which is the difference between the Gaussian-Blurred images at scales σ and $k\sigma$.

The interest points are defined as the extrema (both maxima and minima) in the function $D(x, y, \sigma)$. Each pixel in the DoG image is compared with the pixels of its 26 neighbors to search scale space extrema, where 9 neighbors above and 9 neighbors below the scale and 8 neighbors at the same scale, as is shown in Fig. 3.2. When a pixel is larger or smaller than all its neighbors, it is indicated as a candidate interest point.

The scale space representation of an input image mimics the visual perception at different distances, so the interest points selected by SIFT are scale invariant (Jarrett *et al.*, 2009).

3.2 Key-points localization

After a set of key-point candidates have been collected, those that are sensitive to noise or poorly localized along an edge will be rejected based on measures of their stability. Second order Taylor expansions around key-points are used to localize each of them:

$$D(z_0 + z) \approx D(z_0) + (\frac{\partial D}{\partial z} \mid_{z_0})^T z + \frac{1}{2} z^T (\frac{\partial^2 D}{\partial z^2} \mid_{z_0})^T z, \tag{3.5}$$

where D and its derivatives are evaluated at the key point location $z_0 = [x_0, y_0, \sigma_0]^T$ and z is the offset from this point.

The offset of the extremum, denoted as $\hat{z}$, is computed by taking the derivative of this function with respect to z and setting it to zero:

$$\hat{z} = -(\frac{\partial^2 D}{\partial z^2} \mid_{z_0})^{-1} \cdot (\frac{\partial D}{\partial z} \mid_{z_0}). \tag{3.6}$$

The offset $\hat{z}$ can be estimated using standard difference approximations from neighbors in the DoG. If $\hat{z}$ is larger than 0.5 in any dimension, then it suggests that the extremum lies closer to a different sample point, which is followed by changing the sample point and interpolation instead about that particular point. The final value of $\hat{z}$ is added to the location of its sample point, resulting in getting the interpolated estimated for the location of the extremum. The value at the extremum, denoted as $D(\hat{z})$, is given by substituting equation Eq. 3.6 into Eq. 3.5:

$$D(\hat{x}, \hat{y}, \hat{\sigma}) = D(z_0 + \hat{z}) \approx D(z_0) + \frac{1}{2} (\frac{\partial D}{\partial z} \mid_{z_0})^T \cdot \hat{z}. \tag{3.7}$$

All the values of $\mid D(z_0 + \hat{z}) \mid$ less than a certain threshold are rejected.

Finally, any features located on edges are discarded due to the ambiguity which they may cause if used for matching purposes. A well defined peak is expected to have a large principle curvature across and along the edge.

The principal curvatures can be given from a 2×2 Hessian matrix, H, at the scale and location of the key points:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}. \tag{3.8}$$

The differences of adjacent sample points are used to estimate the derivatives. The principal curvatures of D are proportional to the eigenvalues of H. By taking assumptions that $\lambda_1$is the eigenvalue with the largest magnitude and λ_2 is the smaller one, the sum of the eigenvalues from the trace of H and their product from the determinant can be computed:

$$Tr(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2. \tag{3.9}$$

$$Det(H) = D_{xx} \cdot D_{yy} - (D_{xy})^2 = \lambda_1 \cdot \lambda_2. \tag{3.10}$$

Let r represent the ratio between the largest absolute value of eigenvalue and the smaller one, such that $r = \frac{\lambda_1}{\lambda_2}$:

$$\frac{(Tr(H))^2}{Det(H)} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \cdot \lambda_2} = \frac{(r+1)^2}{r}, \tag{3.11}$$

where $\frac{(r+1)^2}{r}$ is a minimum when the two eigenvalues are equal and it increases with respect to r. Thus we only need to check Eq. 3.12 when we want to make sure that the ratio of principal curvatures is below a certain threshold τ:

$$\frac{(Tr(H))^2}{Det(H)} < \frac{(\tau+1)^2}{\tau}. \tag{3.12}$$

In this thesis, we use a value of $\tau = 10$, which is also used in the standard SIFT method.

Chapter 4

Feature Description and Matching

4.1 Forming the descriptors

Similar to the extraction of SIFT descriptors, the feature descriptors of my proposed method (SINE) and patch-based feature learning, SF, builds a local feature histogram in a small region of the 1^{st} layer or the 2^{nd} layer feature maps around interest points detected by SIFT, which can be seen as a pooling stage (Fig. 4.1).

As is shown in (Rigamonti *et al.*, 2011), although sparsity is important for learning the filters, in our architecture that employs a pooling stage, sparsity is a temporary condition only. The pooling stage is parallel to the complex cells' layers in Hubel and Wiesel's model of the V1 cortex (Hyvrinen *et al.*, 2009) and can provide a certain degree of invariance to minor appearance changes (Rigamonti *et al.*, 2011), which is also a key component of Convolutional Neural Networks proposed by (LeCun *et al.*, 2010). This pooling stage could also reduce the dimensionality of the descriptors from a computational perspective, and thus facilitate subsequent operations (Rigamonti *et al.*, 2011).

In this thesis the pooling stage uses average value pooling. First a $k \times k$ region of a 1^{st} layer or 2^{nd} layer feature map around the interest point is extracted, and I choose $k = 4$ throughout all my experiments in this thesis. The computation of the feature histogram at pixel position $p = (x, y)$ used as the descriptor can be formalized as a vector $F(x, y) \in \mathbb{R}^d$ with

$$F_i(x, y) = Avg_{(x',y')\in P}(\mid z_i(x', y') \mid), \tag{4.1}$$

where P is the set of all pixels within the $k \times k$ region of interest, $\mid z_i(x, y) \mid$ is the absolute value of feature maps at pixel position $p = (x, y)$.

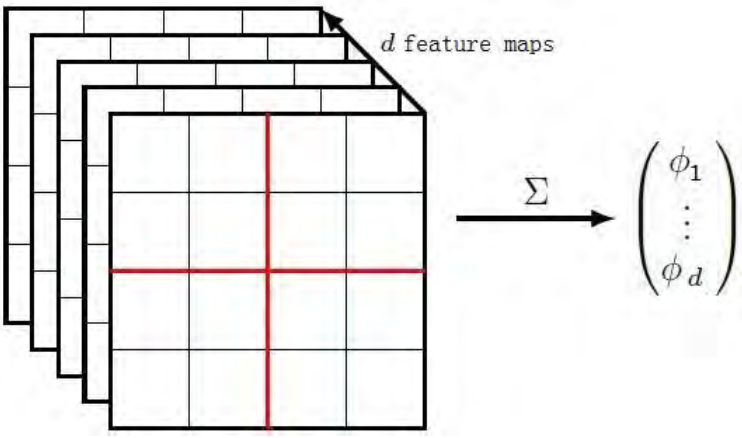


Figure 4.1: Build feature histogram using average value pooling upon the 2^{nd} layer latent feature maps (Blum *et al.*, 2011)

The number of learned latent feature maps d is usually large (in my thesis there are 465 filters in the 2^{nd} layer), so the dimensionality of the descriptor is also high. A PCA over all descriptors can be performed if a smaller feature descriptor is needed (Blum *et al.*, 2011).

4.2 Matching

4.2.1 Tentative matches

After the processes of feature detection and description, each input image is represented as a set of feature vectors around interest points detected by SIFT. Then a nearest neighbor search (NNS) is used to detect the similarities between feature vectors of the two images.

Each feature vector F_i^1 from image 1 is compared to all the feature vectors F_j^2 in image 2 by computing the Euclidean distances $d_{ij}(F_i^1, F_j^2)$.

$$d_{ij}(F_i^1, F_j^2) = \sqrt{\sum_{k=1}^{K}(d_k^1 - d_k^2)^2}, \tag{4.2}$$

where K is the total number of elements in a single feature vector, and in this thesis $K = 465$ for two-layer models and $K = 128$ for one-layer models.

The feature pairs with the minimum Euclidean distances are regarded as potential matches. In order to discard mismatches which arise from background clutter, I apply the Nearest Neighbor Distance Ratio (NNDR) method proposed by Lowe (Lowe, 2004).

For each feature vector from image 1, NNDR compares the Euclidean distances to the nearest and the next nearest neighbor feature vectors of image 2. NNDR discards those matches when their ratio between the nearest and the second nearest distances is below a certain threshold. Because the exhaustive search for the nearest neighbors is computationally expensive, I also use an expanded version of Approximate Nearest Neighbor (ANN), the Best-Bin-First (BBF) proposed by Lowe (Lowe, 2004). BBF searches for bins in feature space in the order of their closest distance from the image 1 and then stop searching when it finishes checking the first 200 nearest neighbors.

4.2.2 Random Sample Consensus

This thesis uses Random Sample Consensus (RANSAC) (Juan and Gwun, 2009) with homography model (Mery, 2011), which is a randomized algorithm for robust parameters estimation problems (Raguram, Frahm, and Pollefeys, 2008), to reject inconsistent matches. First, RANSAC selects a small subset of the input data points and estimates the model parameters from this subset. After this, RANSAC evaluates the model over the entire dataset and determines the number of data points which are consistent with the model.

By doing this repeatedly until the probability of finding a model with more consistent data points than the current best one falls below a predefined threshold (5% in this thesis) (Raguram *et al.*, 2008), we can obtain the inliers and reject outliers at the same time (Juan and Gwun, 2009). In the following experiments, inliers are assumed to represent correct matches.

Chapter 5

Experiments

All the experiments use a MATLAB implementation on a laptop with a 2GHz CPU and 8GB RAM.

5.1 Settings for SF-based feature matching

Experiments are done with 2500 19×19 sample patches randomly taken from two 200×200 images (using grayscale versions of Fig. 5.1 for learning). The SF algorithm is adapted from (Ngiam *et al.*, 2011). The maximum number of iterations until convergence is set to 8000 and the number of learned feature maps[1] is 128. The only pre-processing step is to remove the DC component, which means to subtract the mean of each sample patch from itself (Ngiam *et al.*, 2011). Compared to other patch-based algorithms required to use whitening pre-processing (Ngiam *et al.*, 2011), SF is very efficient.

Fig. 5.2 shows that 2^{nd} layer filters are only larger versions of 1^{st} layer filters, and both of them contain too many similar and unclear filters. By contrast (Fig. 5.3), DN[2] can learn much more diverse, clear and discriminative filters.

When applying SF to 200×200 forest images (shown in Fig. 5.4), the learned filters are random and noisy (shown in Fig. 5.5), which can not be used to form useful descriptors.

It can be seen from Fig. 5.6 that the tentative matches given by SIFT detector and SF descriptor are all incorrect. Therefore I only use DN-based feature matching

[1] Also the number of *filters*

[2] Unlike SF, the number of the 2^{nd} layer filters are controlled by a predefined connectivity matrix, so the number of the 1^{st} filters is 15 and the number of 2^{nd} filters is 120. All the experimental settings are the same as Sec. 5.2.



Figure 5.1: A pair of 200×200 flower images as inputs

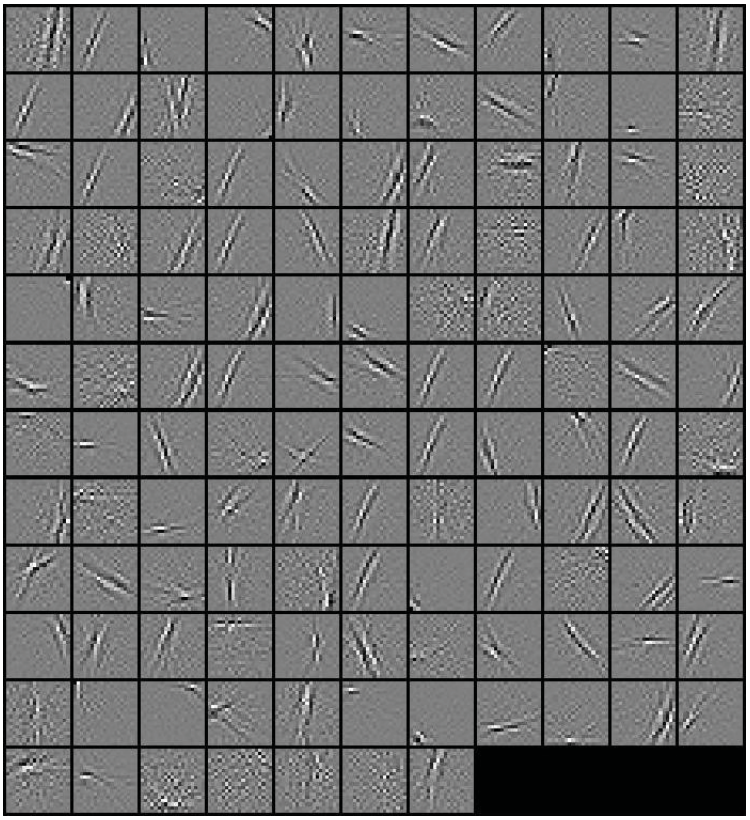
to do further feature matching experiments on aerial images. However, Both of SF[3] and DN succeed in replicating some features of simple cell receptive fields in primary visual cortex (Ranzato *et al.*, 2007)(Olshausen, 1996).

5.2 Settings for DN-based feature matching

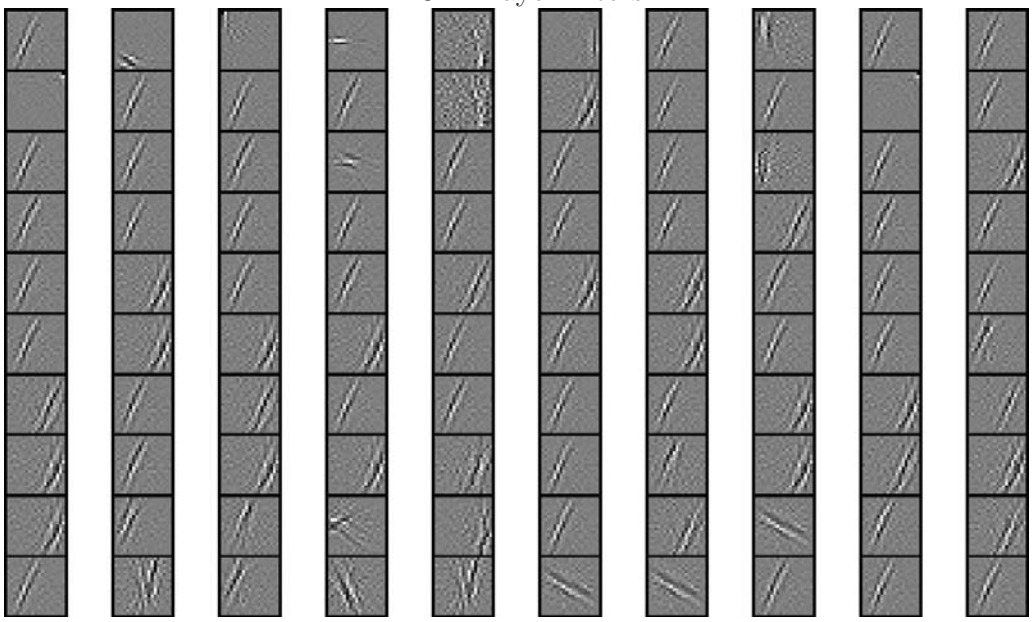
In order to show that my proposed method is able to deal with different kinds of aerial images without any prior knowledge, the testing dataset consists of 8 pairs of 200×200 RGB image patches and their grayscale versions extracted from one pair of 1616×1080 and one pair of 4592×2576 RGB aerial images (taken by an unmanned aerial vehicle and supplied by Areograph Ltd.), of which one containing forest and the other pair containing buildings.

The learning settings throughout this thesis I use are as follows: $H = 19$, $\lambda = 1$, $p = 1$, $\beta_0 = 1$, $\beta_{Inc} = 6$, $\beta_{Max} = 10^5$, $E = 2$. These settings are the same as (Zeiler *et al.*, 2010), except that $H = 7$ and $E = 5$ are used in the original paper. For the one-layer model, the number of the 1^{st} layer feature maps (also the filters) is 128 for each color-channel, which is the same as the number of SIFT descriptors. For the two-layer model, the number of the 1^{st} layer latent feature maps (also the number of filters) is 90 (30 for each color channel), and the number of the 2^{nd} layer latent feature maps is 465.

[3]SF only produces some features of simple cell receptive fields in primary visual cortex on very simple images. The author of SF also indicated through personal communication that in order to speed up the convergence, whitening is an alternative. So one possible cause is that merely removing DC is not enough for the pre-processing procedure. But if using whitening, SF cannot scale well to handle large input dimensions.

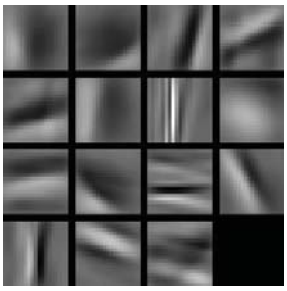


128 1^{st} layer filters

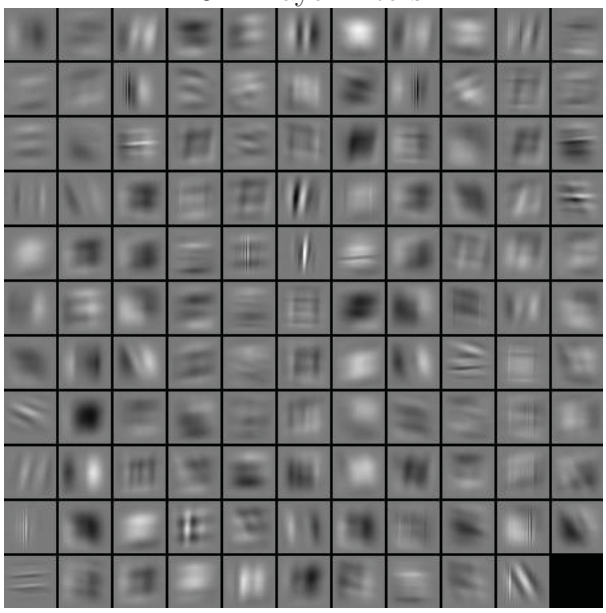


100 2^{nd} layer filters

Figure 5.2: 1^{st} layer and 2^{nd} layer filters learned by SF on flower images



15 1^{st} layer filters



120 2^{nd} filters learned by DN

Figure 5.3: 1^{st} layer and 2^{nd} layer filters learned by DN

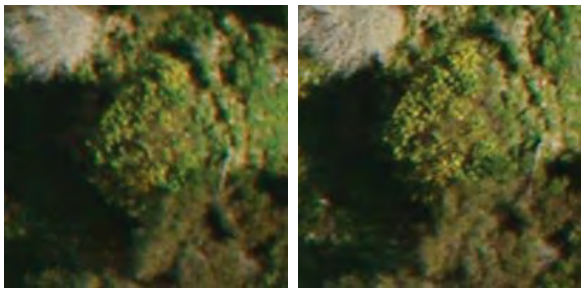
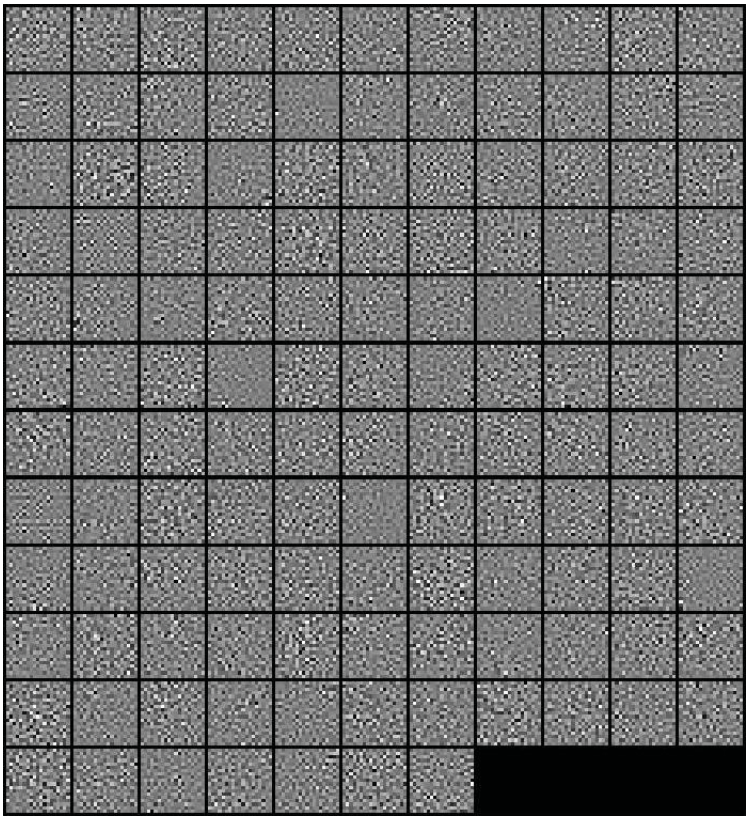
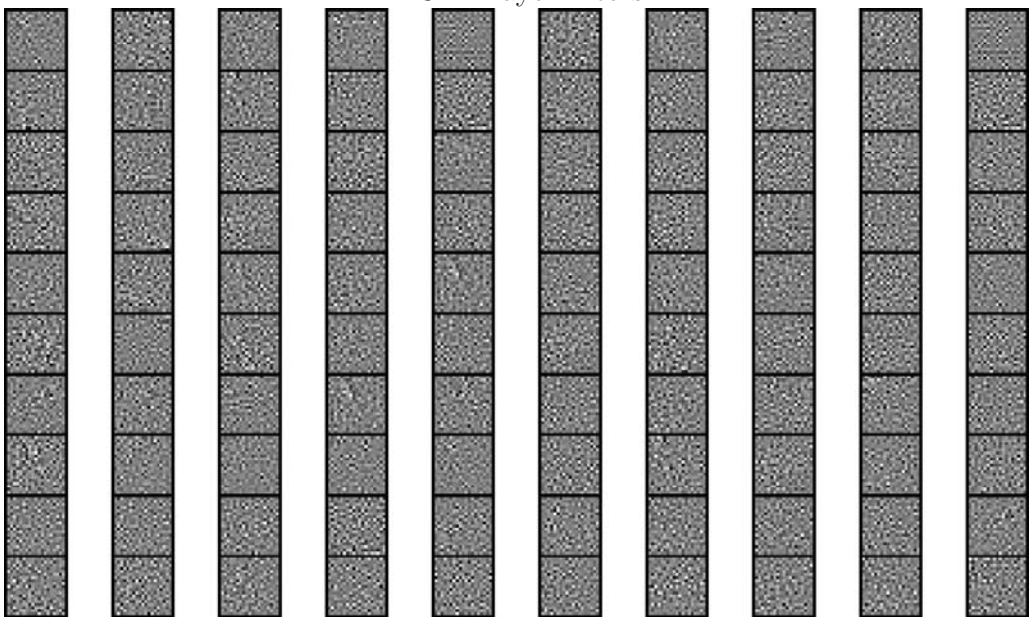


Figure 5.4: A pair of forest images

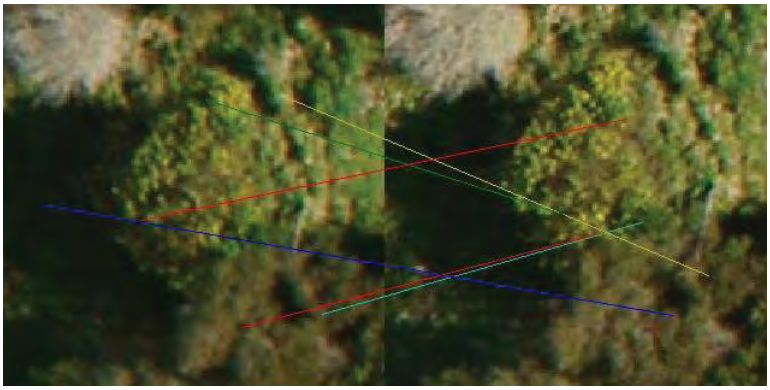


128 1^{st} layer filters



100 2^{nd} layer filters

Figure 5.5: 1^{st} layer and 2^{nd} layer filters learned by SF on forest images



10 tentative matches

Figure 5.6: Matching reslts of using SF on forest images

Through my experiments, when using a two-layer model, 465 2^{nd} layer latent feature maps appear to give the best matching results on aerial datasets. The number of 2^{nd} layer latent feature maps are determined by a connectivity matrix (LeCun, Bottou, Bengio, and Haffner, 1998)(Zeiler *et al.*, 2010) between the 1^{st} layer and the 2^{nd} layer feature maps.

In the following experiments, the number of tentative matches given by SINE and SIFT varies from 15 to 35. It should be noted that different images have different numbers of effective matching points, and the number of effective matches given by SINE is smaller than SIFT's. For instance, if SINE and SIFT are required to match an excessive fixed number of points then SINE is less reliable. Additional experiments without setting a fixed number of tentative matches can be found in the Appendix.

In theory, Deconvolutional Network (DN) is linear in the number of filters and pixels in the image (Zeiler *et al.*, 2010). However, it needs a large amount of RAM to infer enough latent feature maps: if the number of latent feature maps is too small or too big, neither one of the resulting descriptors is discriminative enough. For example, the current implementation requires 2GB RAM to learn 465 2^{nd} layer latent feature maps and approximately 6GB RAM to learn 500 layer-3 latent feature maps from two 200×200 RGB images.

The DN algorithm is adapted from Deconvolutional Networks Toolbox (Zeiler *et al.*, 2010). The SIFT algorithm (including detector and descriptor) is adapted from VLFeat 0.9.14 (Vedaldi and Fulkerson, 2010) and the RANSAC algorithm with homography model is adapted from Balu Toolbox (Mery, 2011) with default settings. The number of tentative matches given by my proposed method (SINE) and SIFT are set to be the same to make a fair comparison.



Figure 5.7: Original aerial images of buildings

5.3 Buildings images

My proposed method (SINE) and the standard way of using SIFT are separately run on 4 pairs of 200×200 RGB image patches and their grayscale versions about buildings extracted from one pairs of 1616×1080 RGB aerial images (Fig. 5.7).

Fig. 5.8 shows the extracted 200×200 RGB patches from original aerial images of buildings.

Fig. 5.9 shows the learned 1^{st} layer filters on buildings images (set 1) when projected into pixel space. It can be seen that the 1^{st} layer filters learn colorful Gabor-style elements.



(a)



(b)



(c)



(d)

Figure 5.8: Extracted patches from aerial images of buildings

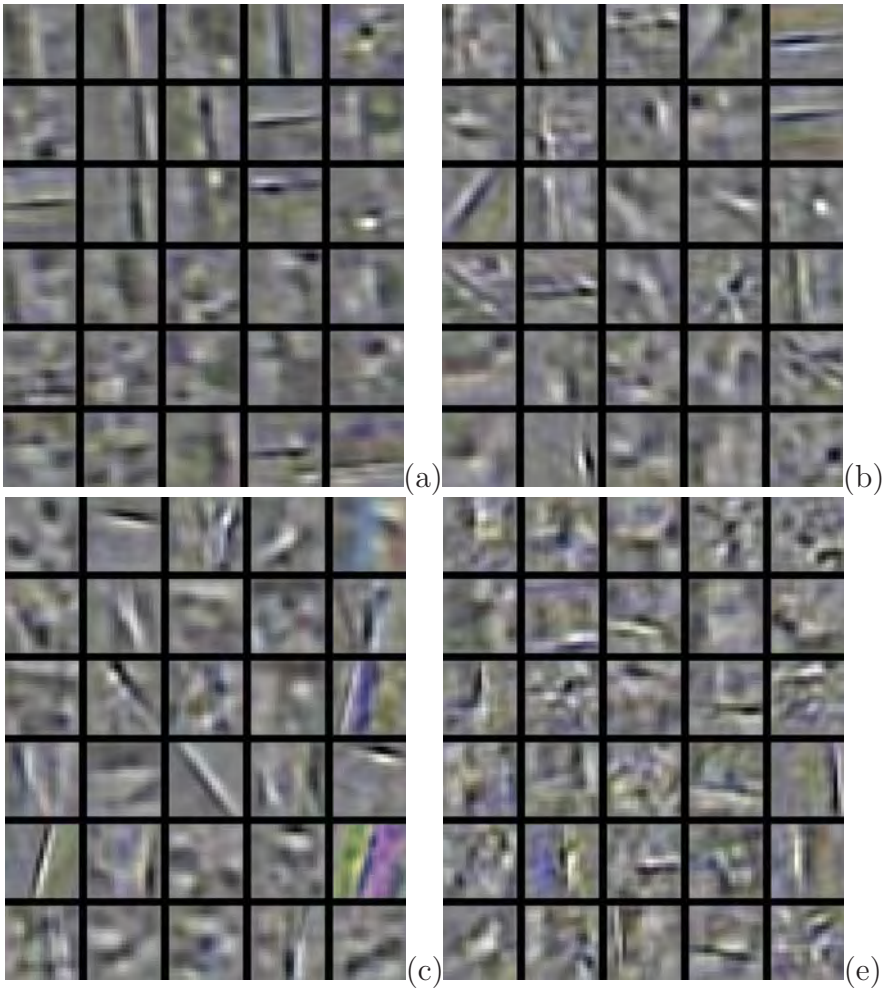


Figure 5.9: 1^{st} layer filters on RGB buildings images projected back into pixel space

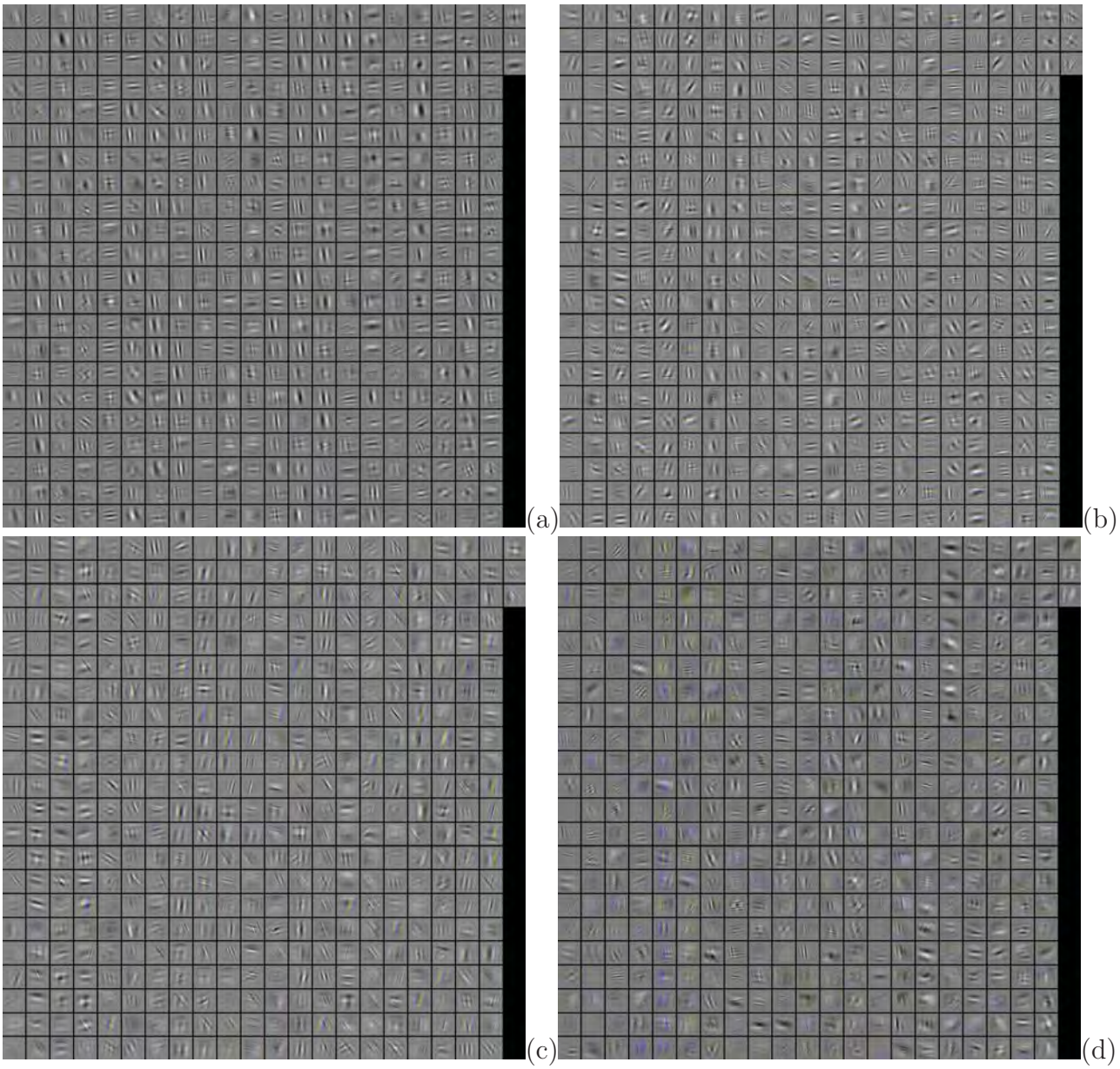


Figure 5.10: 2^{nd} layer filters on RGB buildings images projected back into pixel space

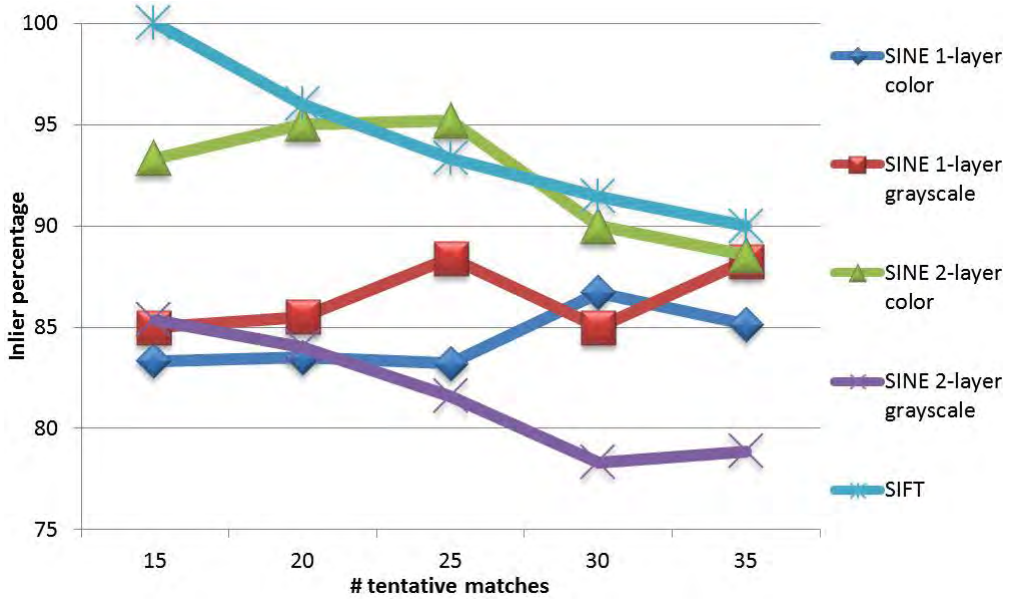


Figure 5.11: Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The first pair of building images.

Fig. 5.10 shows the learned 2^{nd} layer filters on buildings images when projected back into pixel space. It can be seen that the 2^{nd} layer filters are highly diverse and include rich color information. Since building images have regular shapes, many of the them are oriented colorful parallel lines and conjunctions of T-junctions.

SIFT and other prevailing methods are restricted to grayscale images (Manuel Blum, 2011). By contrast, my proposed architecture can make use of color information easily to form a coherent feature representation. Furthermore, it has been shown that (Manuel Blum, 2011) depth information can also be incorporated into the representation and thus increase the recognition performance on the RGB-D object dataset. In this part, I also assess the usefulness of color information.

Fig. 5.11, Fig. 5.12, Fig. 5.13 and Fig. 5.14 show the overall matching results with varying numbers of tentative matches. It can be seen that SINE with two-layer model on RGB images achieves better results consistently than other settings do, so it seems that a deep architecture and color information are indispensable for learning-based feature matching algorithm. However SIFT always gives the best and most consistent results.

SINE's performance is not stable, and one possible cause is that a sparse algorithm

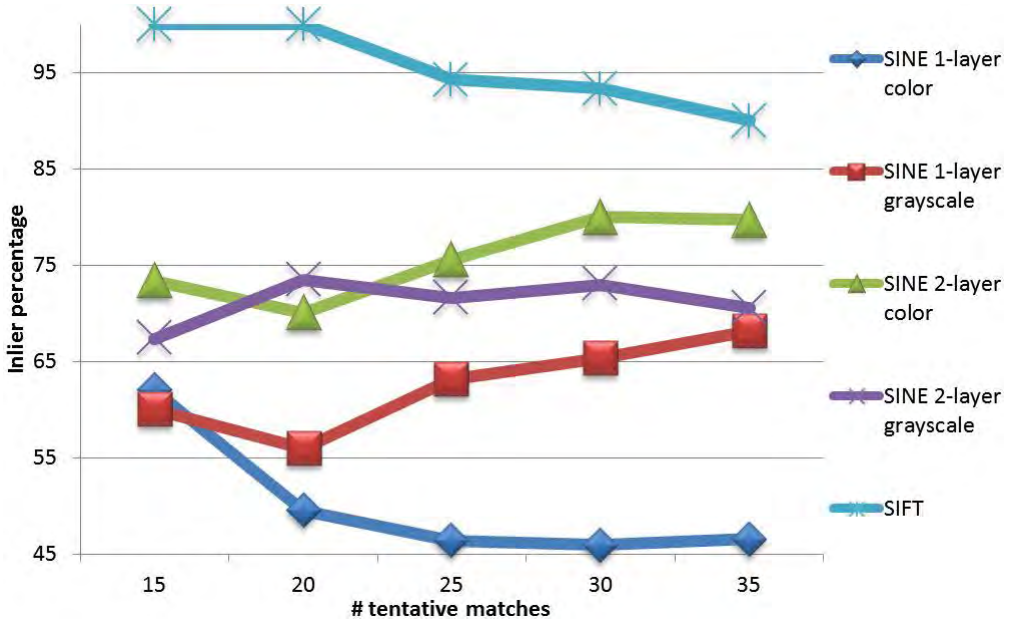


Figure 5.12: Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The second pair of building images. When using SINE with one-layer model on grayscale images, the maximum number of tentative matches is 25.

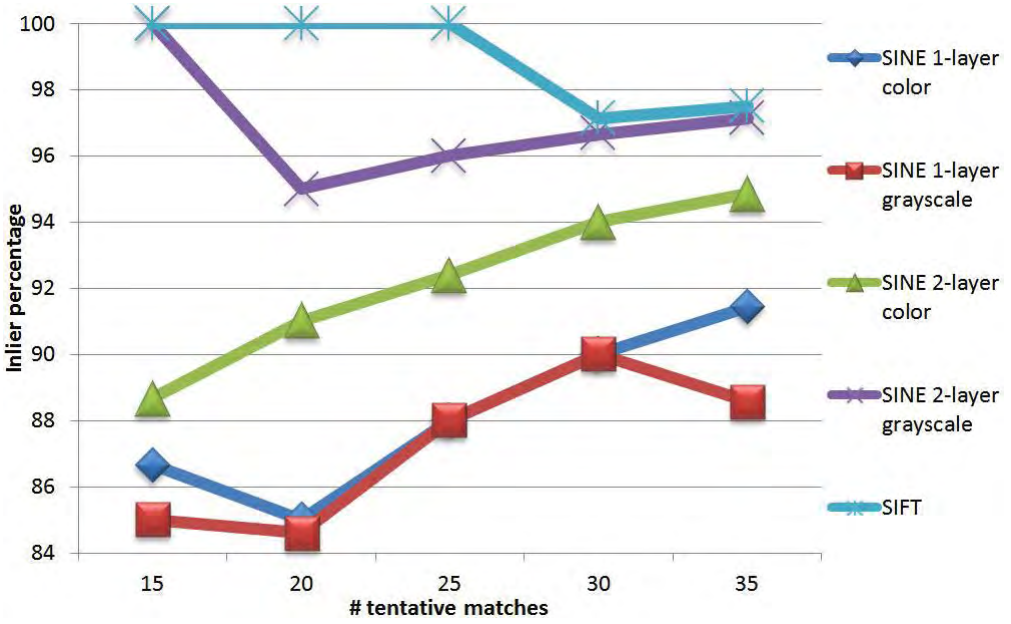


Figure 5.13: Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The third pair of building images.

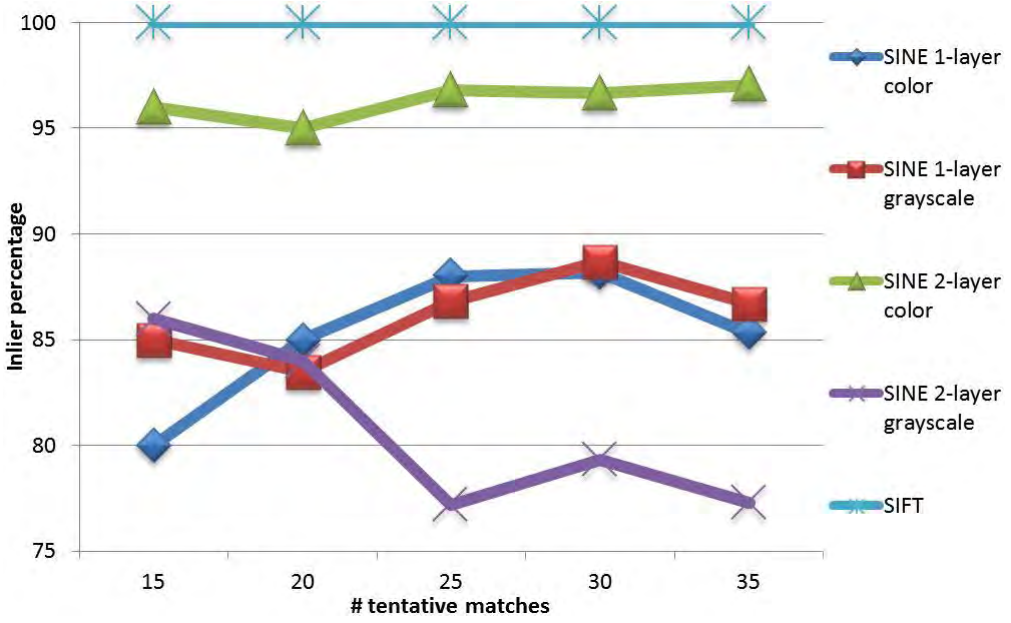


Figure 5.14: Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The forth pair of building images.

can not be stable and vice versa (Xu *et al.*, 2008), where stability means that given two very similar input images, an algorithm's output varies little. Another possibility is that in order to converge sufficiently on large and complicated images[4], the number of epochs E needs to increase correspondingly.

5.4 Forest images

My proposed method (SINE) and the standard way of using SIFT are separately run on 4 pairs of 200×200 RGB image patches and their grayscale versions about forest extracted from one pairs of 1616×1080 RGB aerial images (Fig. 5.15).

Fig. 5.16 shows the extracted 200×200 RGB patches from original aerial images of forest.

Fig. 5.17 shows the learned 1^{st} layer filters on forest images when projected into pixel space. It can be seen that unlike the situations in building images which have regular shapes, the 1^{st} layer filters learn somewhat chaotic colorful elements.

Fig. 5.18 shows the learned 2^{nd} layer filters on forest images (when projected back into pixel space. It can be seen that the 2^{nd} layer filters are highly diverse and include rich color information. Since forest images do not have regular shapes, many of the learned filters are boxes, gratings and blobs.

Fig. 5.19, Fig. 5.20, Fig. 5.21 and Fig. 5.22 show the overall matching results with varying numbers of tentative matches. It can be seen that SINE with two-layer model on RGB images achieves better results consistently than other settings do, so it seems that a deep architecture and color information is indispensable for learning-based feature matching algorithm. However SIFT always gives the best and consistent results.

5.5 Varying filter sizes

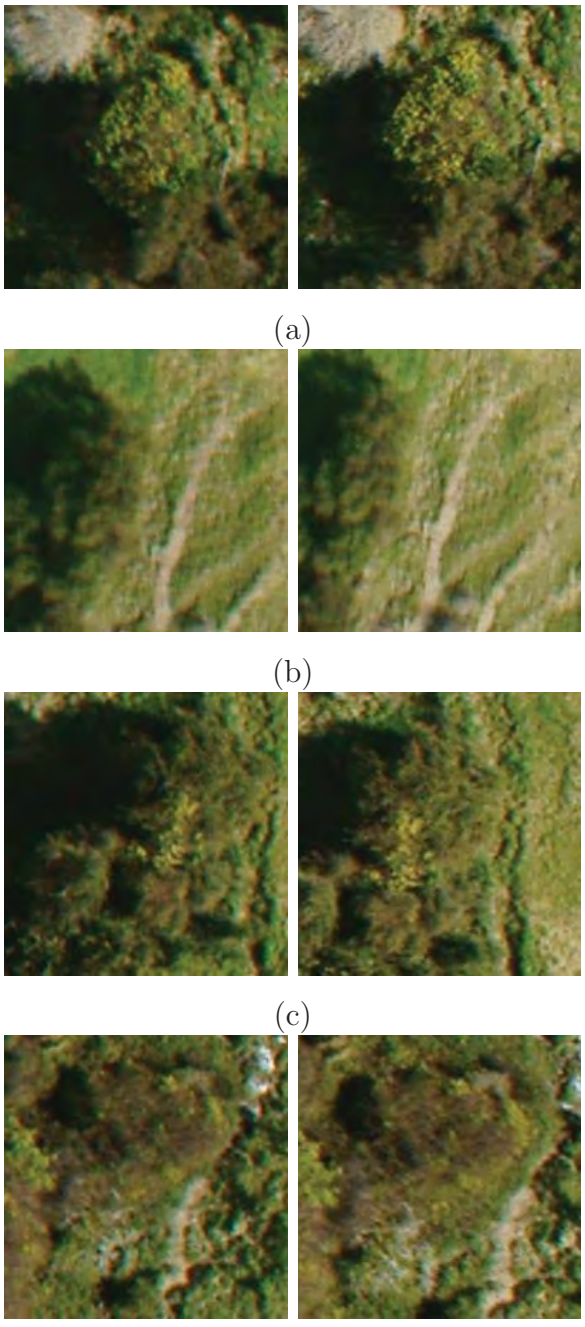
Different filter sizes lead to different feature point matching performances. A series of experiments were run on those four pairs of building images and four pairs of forest images with varying filter[5] sizes ($7 \times 7, 9 \times 9, 11 \times 11, 13 \times 13, 15 \times 15, 17 \times 17, 19 \times 19, 21 \times 21, 23 \times 23$).

[4]The original DN algorithm is run on 150×150 grayscale natural images.

[5]For two-layer models, the 1^{st} and the 2^{nd} layer filter sizes are always the same



Figure 5.15: Original aerial images of forest



(a)

(b)

(c)

(d)

Figure 5.16: Extracted patches from aerial images of forest

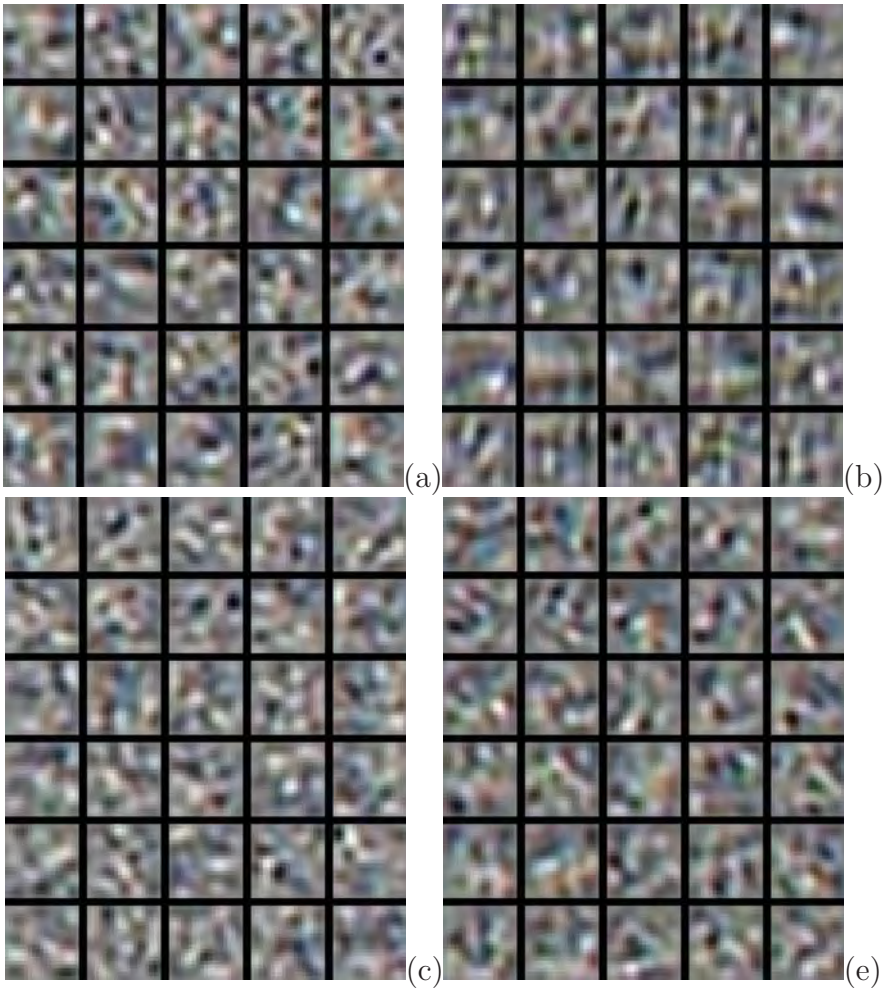


Figure 5.17: 1^{st} layer filters on RGB buildings images projected back into pixel space

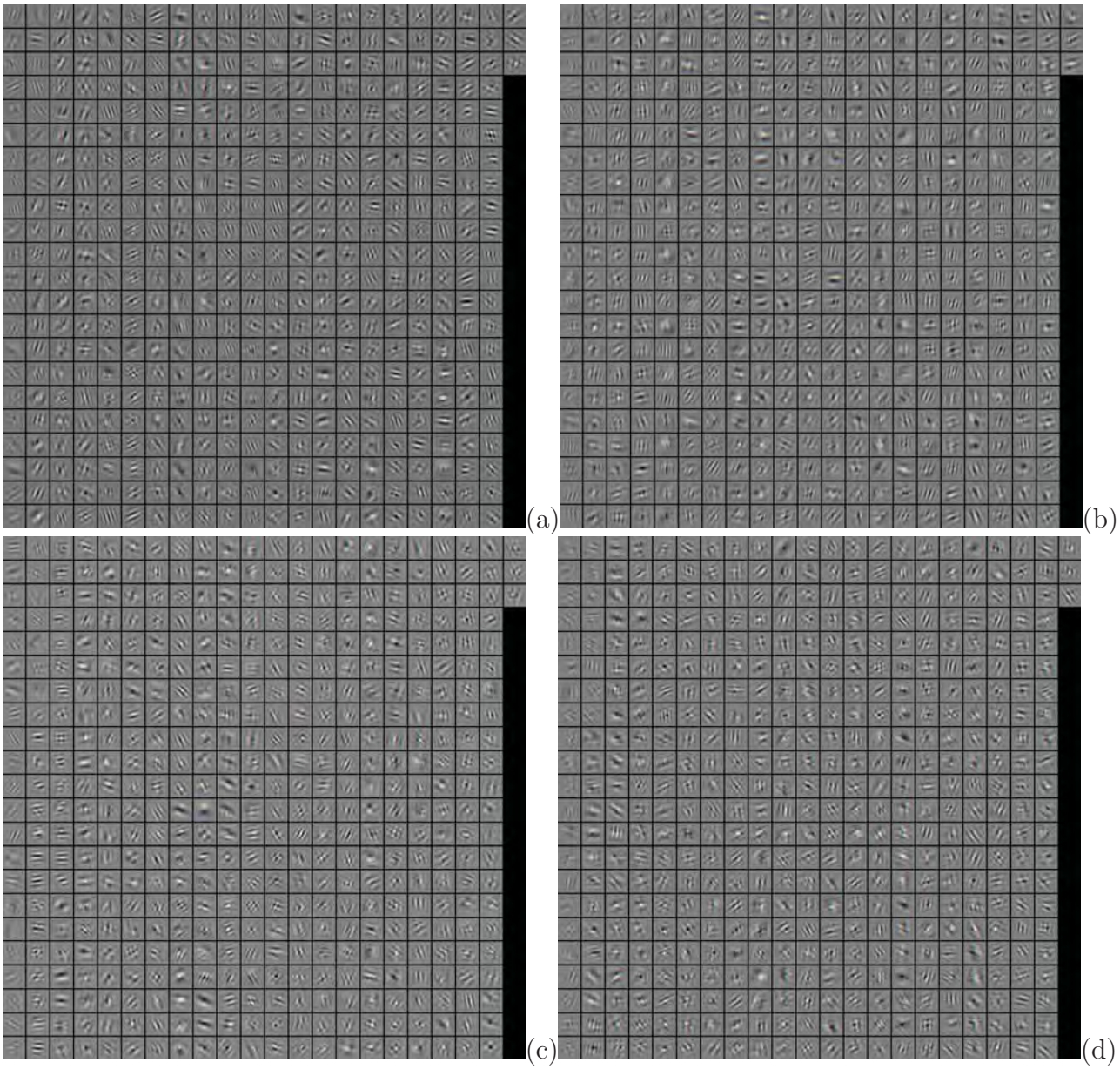


Figure 5.18: 2^{nd} layer filters on RGB buildings images projected back into pixel space

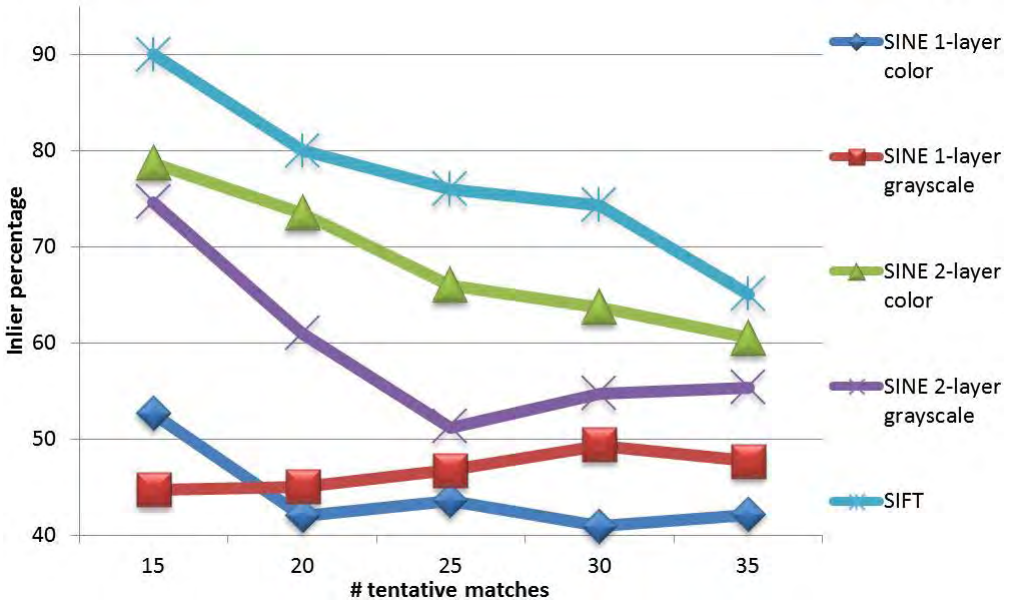


Figure 5.19: Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The first pair of forest images.

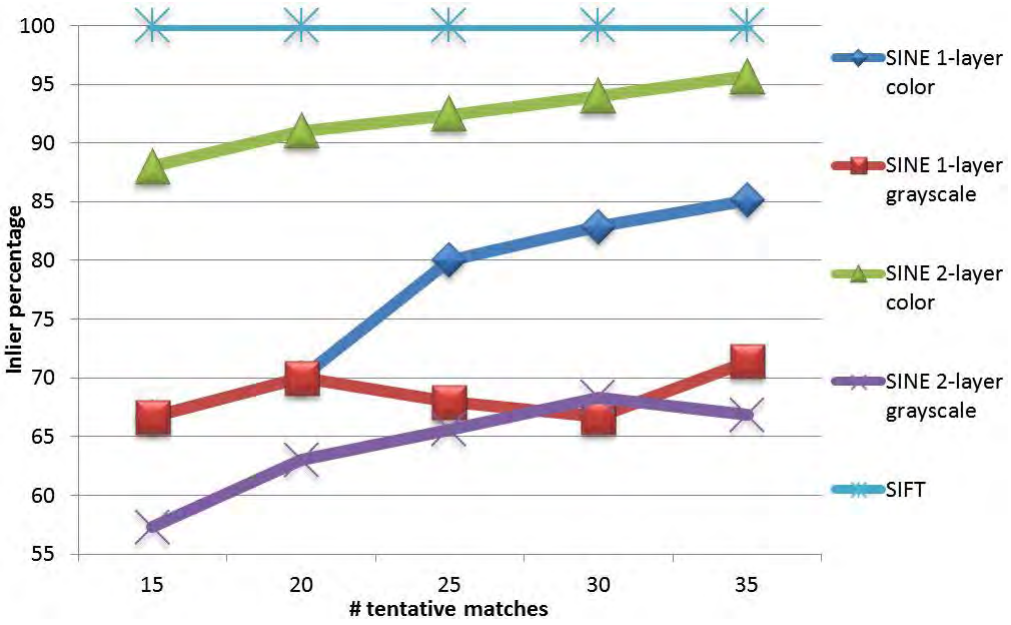


Figure 5.20: Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The second pair of forest images.

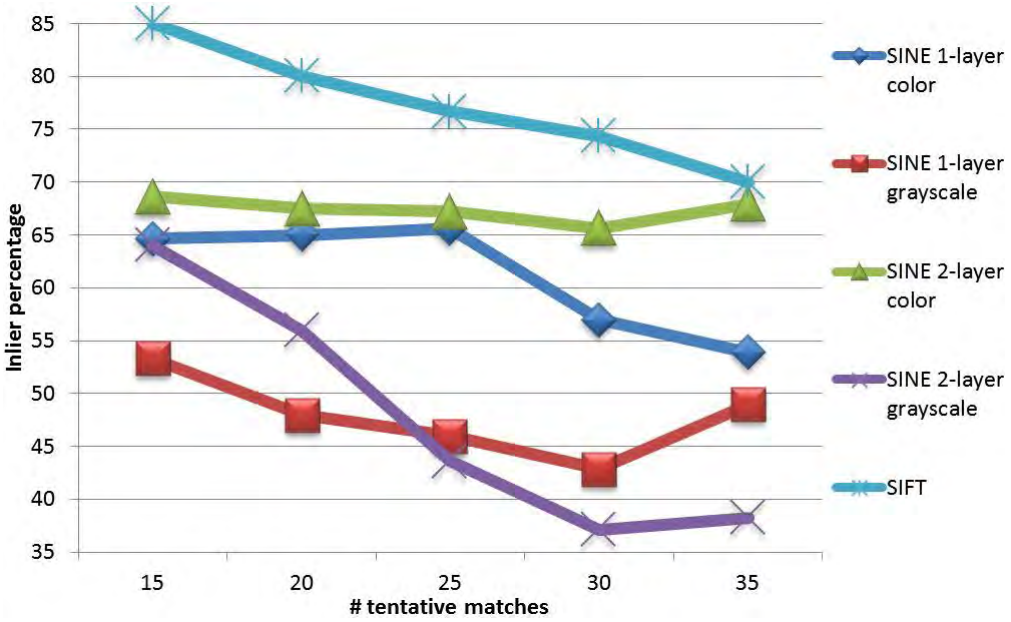


Figure 5.21: Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The third pair of forest images.

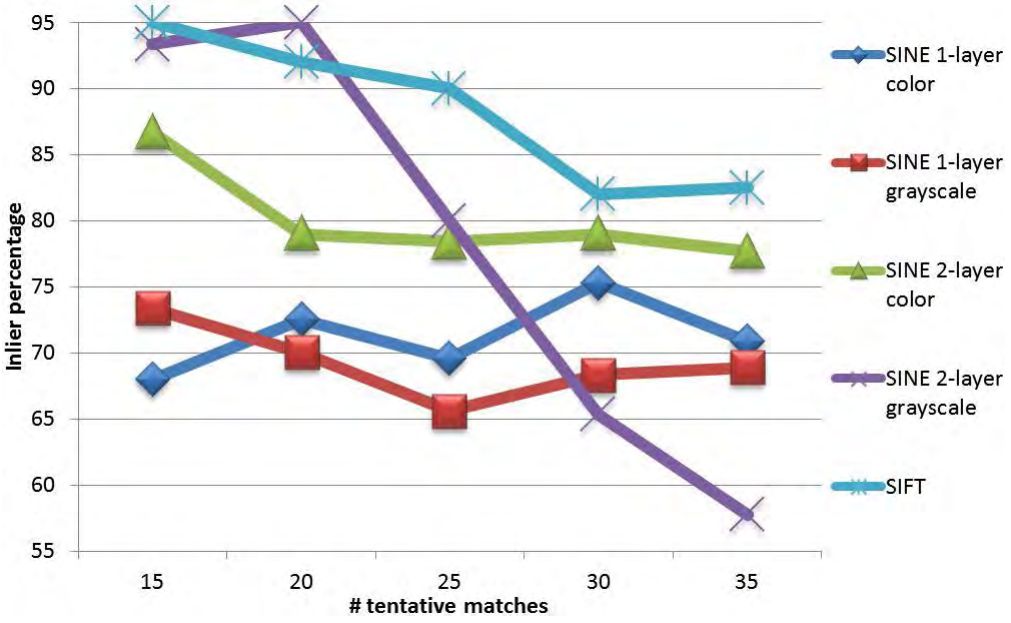
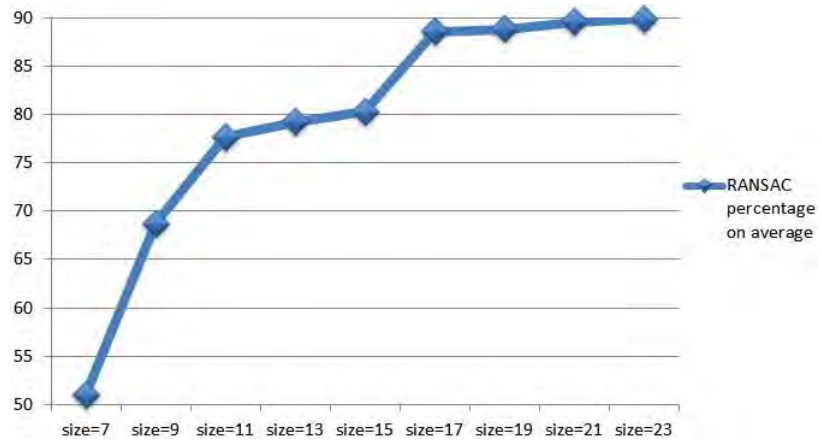
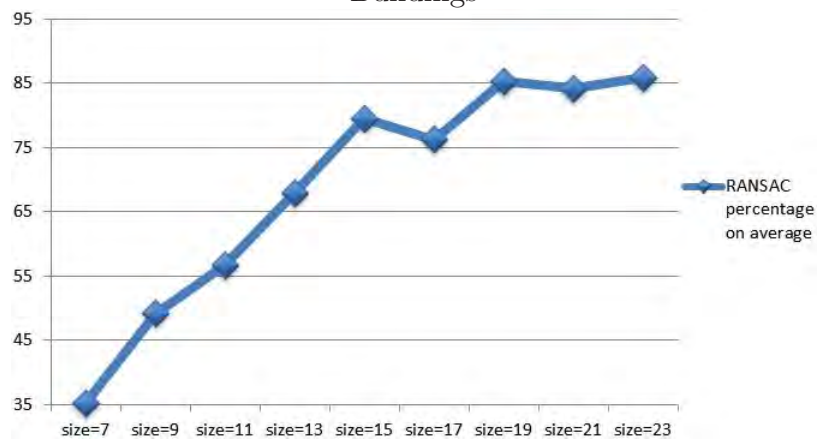


Figure 5.22: Inlier percentage of using my proposed method (including one-layer and two-layer models) on RGB and grayscale images and those that of SIFT method running on grayscale images. The forth pair of forest images.



Buildings



Forest

Figure 5.23: Average percentage of inliers of varying filter sizes on buildings and forest RGB images using two-layer modles

Fig. 5.23 shows that increasing the filter sizes improves the matching performance consistently, and the peaks occur when the filter sizes are above 19×19. However, increasing filter sizes introduces extra boundary effects, resulting in losing the information contained in peripheral areas of input images. And larger filter sizes also require more running time and memory to converge. So the filter size of 19×19 appears to be the best choice for aerial datasets according to matching accuracy, running time and hardware resources.

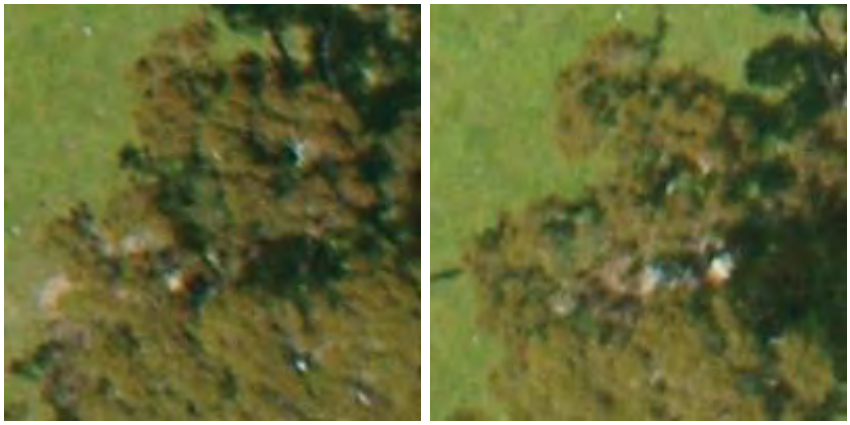


Figure 5.24: A pair of image patches extracted from aerial images of forest

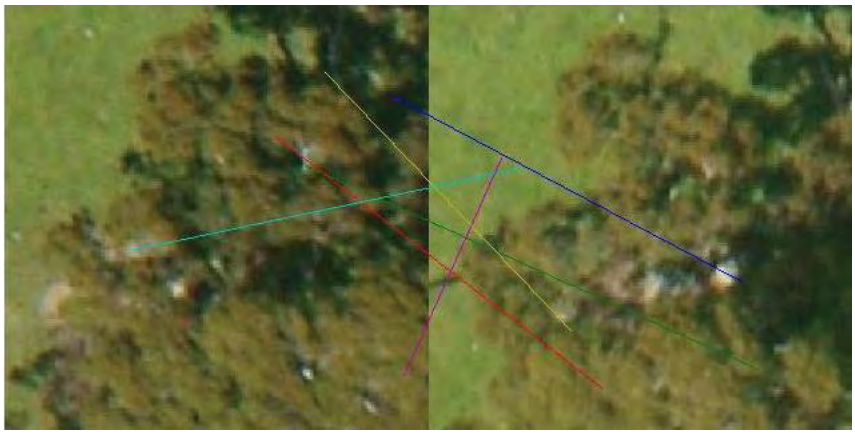


Figure 5.25: 6 tentative matches given by SIFT detector and descriptor

5.6 When SIFT fails

Fig. 5.24 shows a pair of image patches extracted from aerial images of forest, on which SIFT gives totally wrong matches (Fig. 5.25).

By contrast, Fig. 5.26 shows the matching results given by SINE with two-layer model, which is correct. Using SINE with one-layer model also gives correct tentative matches, as shown in Fig. 5.27. It is reasonable to speculate that when applying SINE to larger images (e.g. 400×400), more matching points could be found.

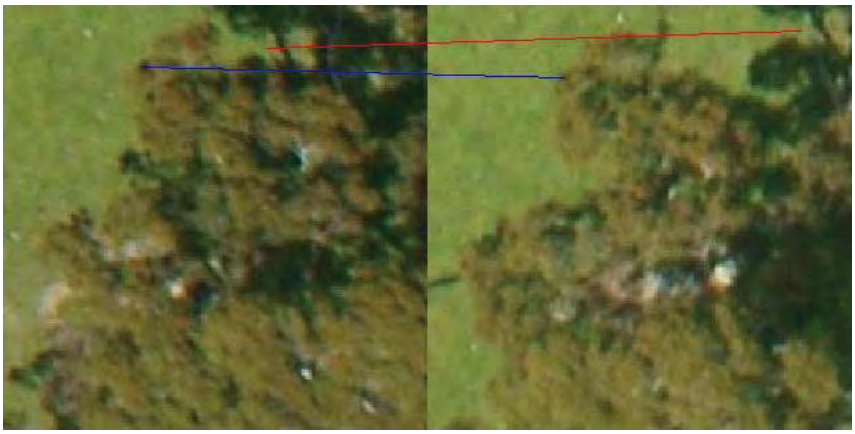


Figure 5.26: 2 tentative matches given by SINE with two-layer model

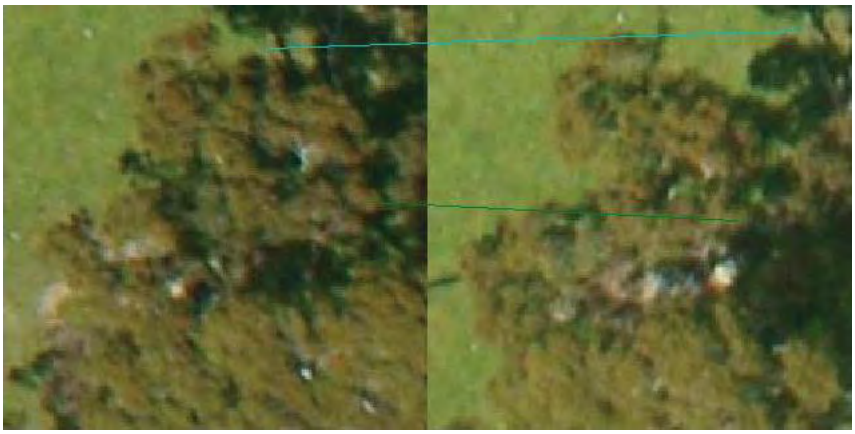


Figure 5.27: 2 tentative matches given by SINE with one-layer model

Chapter 6

Discussion and Outlook

6.1 Relations between the feature detector and descriptor

In this thesis, I focus on combining the standard hand-coded SIFT detector and the trainable DN descriptor to do feature matching tasks. However, the SIFT detector is based on analyzing local gradient properties and only highlights the edge information (Lowe, 2004). Whereas the strength of hierarchical DN model is that it can describe higher-level abstractions which cannot be effectively detected the SIFT edge detector[1].

When dealing with aerial images containing regular shapes (Fig. 6.1) and irregular shapes (Fig. 6.2), interest points chosen by the SIFT detector work best with the two-layer DN model with color information. This may suggest that the regions around these interest points also contain rich higher-level information, such as edge junctions or basic geometric elements, such as rectangles, which can be well described by DN's 2^{nd} layer descriptors. It can also be seen that, those interest points selected by the SIFT detector are along somewhat vague edges in forest images (Fig. 6.2).

From Fig. 6.3, we can see that the 1^{st} layer filters learned from buildings images are similar to Gabor filters, which can detect straight lines very well. However, the 1^{st} layer filters learned from forest images look chaotic, to which SIFT does not fit very well. Therefore, designing a detector which can locate those interest points based on higher-level abstractions, such as edge junctions, will be one of the fundamental questions that we need to address next. Such a detector could be based on saliency

[1] I only use SIFT detector, though other detectors such as Harris detector (Mikolajczyk and Schmid, 2002) could potentially be used.



Applying two-layer model to RGB images

Figure 6.1: Matching results after using RANSAC based on SINE on building images

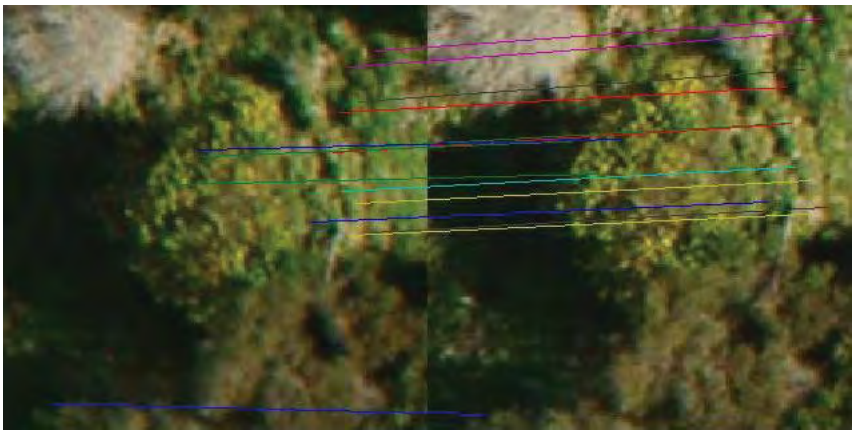
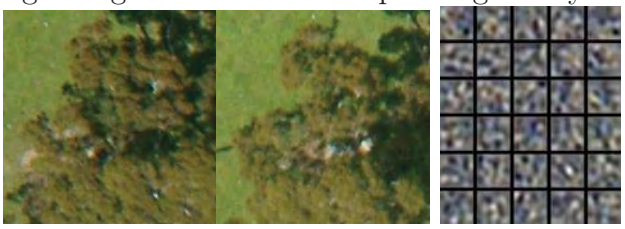


Figure 6.2: Matching results after using RANSAC based on SINE on RGB forest images



buildings images and their corresponding 1^{st} layer filters



Forest images and their corresponding 1^{st} layer filters

Figure 6.3: 1^{st} layer filters learned by Deconvolutional Networks on buildings and forest

maps (Kanan and Cottrell, 2010), which can be either linearly or non-linearly generated from the learned feature representations.

Furthermore, unlike the SIFT descriptor, my proposed descriptors are not rotation and scale invariant[2]. Future work will also focus on learning rotation and scale invariant descriptors in order to improve the feature matching performance in real world applications.

[2]The normalized penalty (i.e., $\| e \|_1 \ / \ \| e \|_2$) used by Sparse Filtering (SF) is scale invariant (Ngiam *et al.*, 2011)

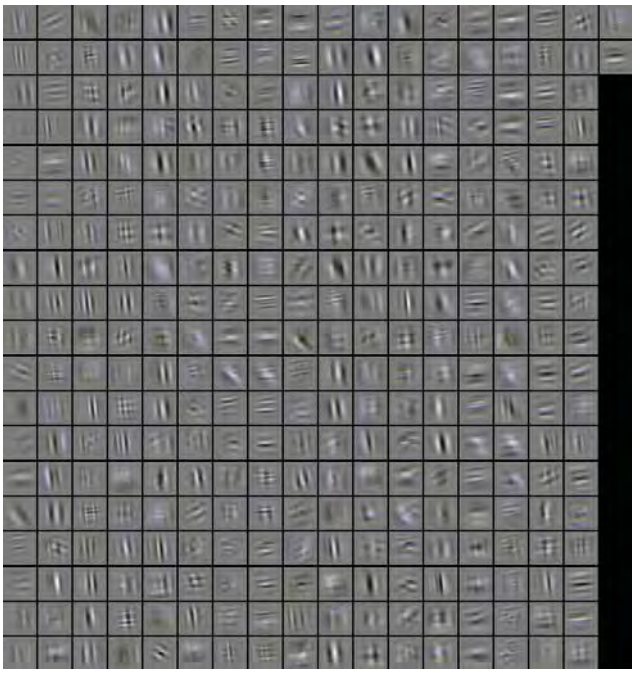
6.2 Why only 2 layers?

Experiments on building and forest images show that SINE with two-layer model on RGB images achieves better results consistently than other settings do, so it seems that a deep architecture and color information are very useful for learning-based feature matching algorithms.

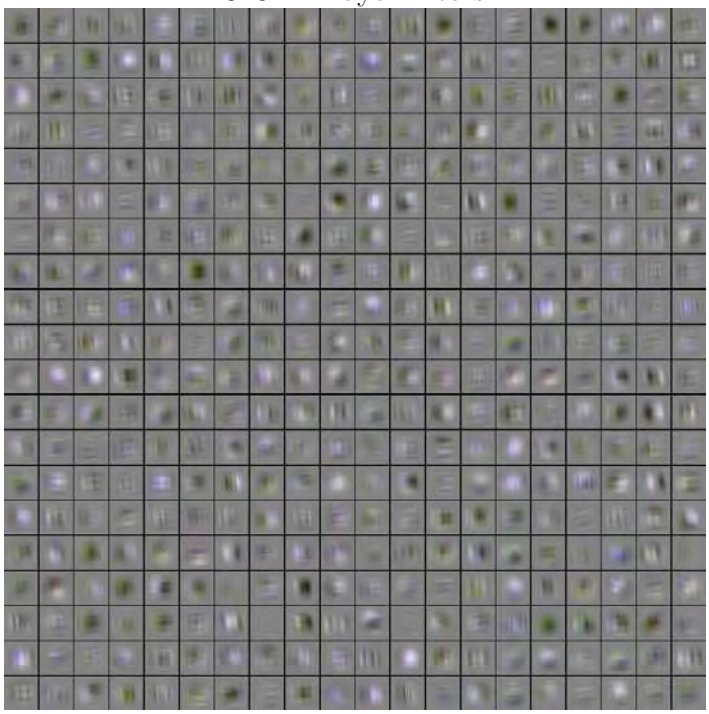
Like other hierarchical feature learning algorithms (Lee *et al.*, 2009)(Kavukcuoglu *et al.*, 2010), DN is trained greedily from bottom up by using the output of the lower layer as input for the next (Zeiler *et al.*, 2011). When the number of layers of learning models is relatively small (e.g. 2 layers), the learning is robust enough to do subsequent point matching task. However, because the input images are lost after the first layer, the subsequent layers of the learning architecture dilute the connection to the input inevitably, thus making the learning fragile (Zeiler *et al.*, 2011). In (Zeiler *et al.*, 2011), a set of latent switch variables are introduced such that the learned filters are adapted to the observed data, thus providing a direct path to the input. In other words, every layer in the model are trained with respect to the original images. However, this technique cannot be easily applied to feature matching. Because while the learned filters are shared between images, the switch variables are not, which means that the feature representations of two input images are not directly comparable. Thus, how to construct a multi-layer model without diluting the connection to the input and how to use it to do feature matching are open questions.

The number of correct matches of using layer-3 latent feature maps obtained from a set of aerial images is always zero in my experiments. By looking at Fig. 6.4, we can clearly see that almost half of the layer-3 filters are becoming too vague to be distinguishable from each other, though some of them are more complex than those in 2^{nd} layer.

Fig. 6.5 shows that three-layer models give very poor matching results.



325 2^{nd} layer filters



400 layer-3 filters

Figure 6.4: 2^{nd} layer and 3^{rd} layer filters learned from two 200×200 images of buildings. Due to the limitation of the RAM (8GB), the number of filters in each layers are 325 and 400 respectively rather than 30 and 465 used in the former experiments with 2 layers models. This figure is best viewed in electronic form.

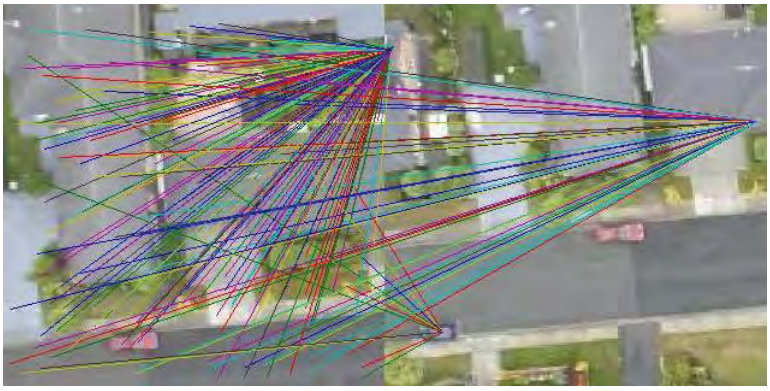


Figure 6.5: 186 tentative matches using the layer-3 filters on RGB building images

6.3 Speed

It takes approximately 40 minutes to learn 465 2^{nd} layer latent feature maps from 2 input images. Although my proposed matching architecture is slow compared to standard SIFT approach (SIFT needs not to learn at all), it is possible to get 1 or 2 orders of magnitude speed-up by using GPU parallel computing due to DN's heavy use of convolution operations (Zeiler *et al.*, 2010). Faster speed also enables us to increase the number of convergence epochs accordingly, which may potentially increase the performance of learned features.

Because the process of learning filters is the most time consuming part of DN, a highly desirable alternative would be to to pre-train filters on a dataset and then use these filters to generate latent feature maps from new instances, which have been shown successful in scene classification tasks (LeCun *et al.*, 2010)(Ngiam *et al.*, 2011)(Zeiler *et al.*, 2010)(Zeiler *et al.*, 2011). In order to learn a representative bank of filters, it is required to run DN on a reasonably large set of images (e.g. 3060 images in Caltech-101 dataset (Fei-Fei, Fergus, and Perona, 2007)) simultaneously, because each filter has to compete with all the others. However this requires huge amount of memory (two 200×200 RGB images would need 1.7GB RAM) and running time. Therefore, I did not study transfer learning in this thesis but it is worth studying further. It should also be noted that my proposed method (SINE) itself is not fixed to any specific dataset, and can be applied to any dataset with no prior knowledge.

Chapter 7

Conclusion

This thesis has explored the problems of feature point matching by applying the SIFT detector and convolutionally-learned feature representations with one or two layers to a pair of RGB aerial images and their grayscale versions, which captures low- and mid-level image representations in a biologically-inspired and unsupervised fashion. To the best of my knowledge, this is the first time that feature learning has been used for feature point matching in the literature.

Although experimental results show that the SIFT method (including the detector and the descriptor) always gives the best and most consistent results, it can also be seen that my proposed hybrid biologically-inspired architecture using two-layer convolution-based models achieves reasonable performance in real-world feature point matching tasks on RGB aerial images without hand-encoding any prior knowledge about the input images. Therefore, I conclude that a learned feature descriptor is a promising tool for challenging feature point matching tasks.

References

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, *110*(3), 346 – 359.

Behnke, S. (2003). Hierarchical Neural Networks For Image Interpretation. *Lecture Notes In Computer Science*, *2766*, 1–13.

Belhumeur, P., Hespanha, J., and Kriegman, D. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(7), 711–720.

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy Layer-wise Training of Deep Networks. In *Advances in Neural Information Processing Systems*, 153–160.

Blum, M., Springenberg, J. T., Wulfing, J., and Riedmiller, M. (2011). On the Applicability of Unsupervised Feature Learning for Object Recognition in RGB-D Data. Downloaded from http://deeplearningworkshopnips2011.files.wordpress.com/2011/12/26.pdf on 6th July 2012. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Bo, L., Ren, X., and Fox, D. (2010). Kernel Descriptors for Visual Recognition. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Eds.), *Advances in Neural Information Processing Systems*, 244–252.

Bo, L., Ren, X., and Fox, D. (2011). Depth Kernel Descriptors for Object Recognition. In *IEEE International Conference on Intelligent Robots and Systems*, 821–826.

Boureau, Y., Bach, F., LeCun, Y., Ponce, J., *et al.* (2010). Learning Mid-level Features for Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2559–2566.

Bouvrie, J. (2009). *Hierarchical Learning: Theory with Applications in Speech and Vision*. Ph. D. thesis, Massachusetts Institute of Technology.

Boynton, G., Demb, J., Glover, G., and Heeger, D. (1999). Neuronal Basis of Contrast Discrimination. *Vision research*, *39*(2), 257–269.

Brady, N. and Field, D. (2000). Local Contrast in Natural Images: Normalisation and Coding Efficiency. *Perception*, *29*(9), 1041–1055.

Chen, B., Polatkan, G., Sapiro, G., Dunson, D., and Carin, L. (2011). The Hierarchical Beta Process for Convolutional Factor Analysis and Deep Learning. In L. Getoor and T. Scheffer (Eds.), *International Conference on Machine Learning*, New York, NY, USA, 361–368. ACM.

Cheung, W. and Hamarneh, G. (2007). N-sift: N-dimensional Scale Invariant Feature Transform for Matching Medical Images. In *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 720–723.

Coates, A., Lee, H., and Ng, A. (2010). An Analysis of Single-layer Networks In Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, Volume 1001, 48109.

Coates, A. and Ng, A. (2011). The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization. In L. Getoor and T. Scheffer (Eds.), *International Conference on Machine Learning*, 921–928.

Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, Volume 1, 886–893.

Elad, M. and Aharon, M. (2006). Image Denoising via Sparse and Redundant Representations over Learned Dictionaries. *IEEE Transactions on Image Processing*, *15*(12), 3736–3745.

Fei-Fei, L., Fergus, R., and Perona, P. (2007). Learning Generative Visual Models From Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. *Computer Vision and Image Understanding*, *106*(1), 59–70.

Field, D. (1994). What Is The Goal of Sensory Coding? *Neural computation*, *6*(4), 559–601.

Gao, S., Tsang, I., Chia, L., and Zhao, P. (2010). Local Features Are Not Lonely–Laplacian Sparse Coding for Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3555–3561.

Geman, D. and Yang, C. (1995). Nonlinear Image Recovery with Half-quadratic Regularization. *IEEE Transactions on Image Processing*, *4*(7), 932–946.

Hawkins, J. and George, D. (2006). Hierarchical Temporal Memory: Concepts, Theory and Terminology. Technical report, Numenta Inc.

Hinton, G. (2007). Learning Multiple Layers of Representation. *Trends in cognitive sciences*, *11*(10), 428–434.

Hinton, G., Osindero, S., and Teh, Y. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*, *18*(7), 1527–1554.

Hinton, G. and Salakhutdinov, R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504.

Hyvärinen, A., Karhunen, J., and Oja, E. (2001). *Independent Component Analysis*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. J. Wiley.

Hyvrinen, A., Hurri, J., and Hoyer, P. O. (2009). *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision* (1st ed.). Springer Publishing Company, Incorporated.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What Is The Best Multi-stage Architecture for Object Recognition? In *IEEE International Conference onComputer Vision*, 2146–2153.

Johnson, A. and Hebert, M. (1999). Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(5), 433–449.

Jolliffe, I. and MyiLibrary (2002). *Principal Component Analysis*, Volume 2ed. Springer.

Juan, L. and Gwun, O. (2009). A Comparison of SIFT, PCA-SIFT and SURF. *International Journal of Image Processing*, *3*(4), 143–152.

Kanan, C. and Cottrell, G. (2010). Robust Classification of Objects, Faces, and Flowers Using Natural Image Statistics. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2472–2479.

Kavukcuoglu, K., Sermanet, P., Boureau, Y., Gregor, K., Mathieu, M., and LeCun, Y. (2010). Learning Convolutional Feature Hierarchies for Visual Recognition. In *Advances in Neural Information Processing Systems*.

Krishnan, D. and Fergus, R. (2009). Fast Image Deconvolution Using Hyper-Laplacian Priors. In *Advances in Neural Information Processing Systems*, Volume 22, 1–9.

Krizhevsky, A. and Hinton, G. (2009). Learning Multiple Layers of Features From Tiny Images. Master's thesis, University of Toronto.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, Volume 2, 2169–2178.

Le, Q., Karpenko, A., Ngiam, J., and Ng, A. (2011). ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning. In *Advances in Neural Information Processing Systems*, 1017–1025.

Le, Q., Ngiam, J., Chen, Z., hao Chia, D. J., Koh, P. W., and Ng, A. (2010). Tiled Convolutional Neural Networks. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta (Eds.), *Advances in Neural Information Processing Systems*, 1279–1287.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional Networks and Applications in Vision. In *IEEE International Symposium on Circuits and Systems*, 253–256.

Lee, H. (2010). *Unsupervised Feature Learning Via Sparse Hierarchical Representations*. Ph. D. thesis, Stanford University.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient Sparse Coding Algorithms. In B. Schölkopf, J. Platt, and T. Hoffman (Eds.), *Advances in Neural Information Processing Systems*, Cambridge, MA, 801–808. MIT Press.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. (2009). Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *International Conference on Machine Learning*, 609–616.

Leung, T. and Malik, J. (2001). Representing and Recognizing the Visual Appearance of Materials Using Three-dimensional Textons. *International Journal of Computer Vision*, *43*(1), 29–44.

Lowe, D. (1999). Object Recognition from Local Scale-invariant Features. In *IEEE International Conference on Computer Vision*, Volume 2, 1150–1157.

Lowe, D. (2004). Distinctive Image Features from Scale-invariant Keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Mairal, J. (2010). *Sparse Coding for Machine Learning, Image Processing and Computer Vision*. Ph. D. thesis, Ecole Normale Superieure.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2009). Non-local Sparse Models for Image Restoration. In *IEEE International Conference on Computer Vision*, 2272–2279.

Mery, D. (2011). BALU: A Toolbox Matlab for Computer Vision, Pattern Recognition and Image Processing (http://dmery.ing.puc.cl/index.php/balu).

Mikolajczyk, K. and Schmid, C. (2002). An Affine Invariant Interest Point Detector. In *The European Conference on Computer Vision*, 128–142. Springer.

Ngiam, J., Koh, P. W., Chen, Z., Bhaskar, S., and Ng, A. (2011). Sparse Filtering. In *Advances in Neural Information Processing Systems*, 1125–1133.

O'Hara, S. and Draper, B. A. (2011). Introduction to the Bag of Features Paradigm for Image Classification and Retrieval. *arXiv*, *abs/1101.3354*.

Olshausen, B. (1996). Emergence of Simple-cell Receptive Field Properties by Learning A Sparse Code for Natural Images. *Nature*, *381*(6583), 607–609.

Olshausen, B. and Field, D. (1997). Sparse Coding with An Overcomplete Basis Set: A Strategy Employed By V1? *Vision research*, *37*(23), 3311–3325.

Raguram, R., Frahm, J., and Pollefeys, M. (2008). A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-time Random Sample Consensus. In *The European Conference on Computer Vision*, 500–513. Springer.

Ranzato, M., Huang, F., Boureau, Y., and LeCun, Y. (2007). Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

Rigamonti, R., Brown, M., and Lepetit, V. (2011). Are Sparse Representations Really Relevant for Image Classification? In *IEEE Conference on Computer Vision and Pattern Recognition*, 1545–1552.

Se, S., Lowe, D., and Little, J. (2002). Mobile Robot Localization and Mapping with Uncertainty Using Scale-invariant visual landmarks. *The International Journal of Robotics Research*, *21*(8), 735–758.

Serre, T., Wolf, L., and Poggio, T. (2005). Object Recognition with Features Inspired by Visual Cortex. In *IEEE Conference on Computer Vision and Pattern Recognition*, Volume 2, 994–1000.

Stone, J. (2004). *Independent Component Analysis: A Tutorial Introduction*. The MIT Press.

Vedaldi, A. and Fulkerson, B. (2010). VLFeat: An Open And Portable Library of Computer Vision Algorithms. In *International Conference on Multimedia*, 1469–1472.

Wallisch, P. (2009). *MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB*. Academic Press.

Wang, Y., Yang, J., Yin, W., and Zhang, Y. (2008). A New Alternating Minimization Algorithm for Total Variation Image reconstruction. *SIAM Journal on Imaging Sciences*, *1*(3), 248–272.

Willmore, B. and Tolhurst, D. (2001). Characterizing the Sparseness of Neural Codes. *Network: Computation in Neural Systems*, *12*(3), 255–270.

Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T., and Yan, S. (2010). Sparse Representation for Computer Vision and Pattern Recognition. *Proceedings of the IEEE*, *98*(6), 1031–1044.

Xu, H., Mannor, S., and Caramanis, C. (2008). Sparse Algorithms Are Not Stable: A No-free-lunch Theorem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*, 187–193.

Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1794–1801.

Zeiler, M., Krishnan, D., Taylor, G., and Fergus, R. (2010). Deconvolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2528–2535.

Zeiler, M., Taylor, G., and Fergus, R. (2011). Adaptive Deconvolutional Networks for Mid and High Level Feature Learning. In *IEEE International Conference on Computer Vision*, 2018–2025.

Zhou, X., Zhuang, X., Tang, H., Hasegawa-Johnson, M., and Huang, T. (2008). A Novel Gaussianized Vector Representation for Natural Scene Categorization. In *International Conference on Pattern Recognition*, 1–4.

Appendix A

Additional Experiments

Both my proposed method (SINE) and SIFT are run on 8 sets of aerial image patches, and every set consists of 4 pairs of patches. Throughout these additional experiments, the SINE uses a two-layer model to learn 465 19×19 filters, and the number of tentative matches given by SINE and SIFT are set to be the same. All the other settings are the same as Chapter 5.

A.1 Rocks

A.1 shows the pair of original 1616×1080 RGB aerial images of rocks.

A.2 shows the extracted 200×200 RGB patches from original aerial images of rocks.

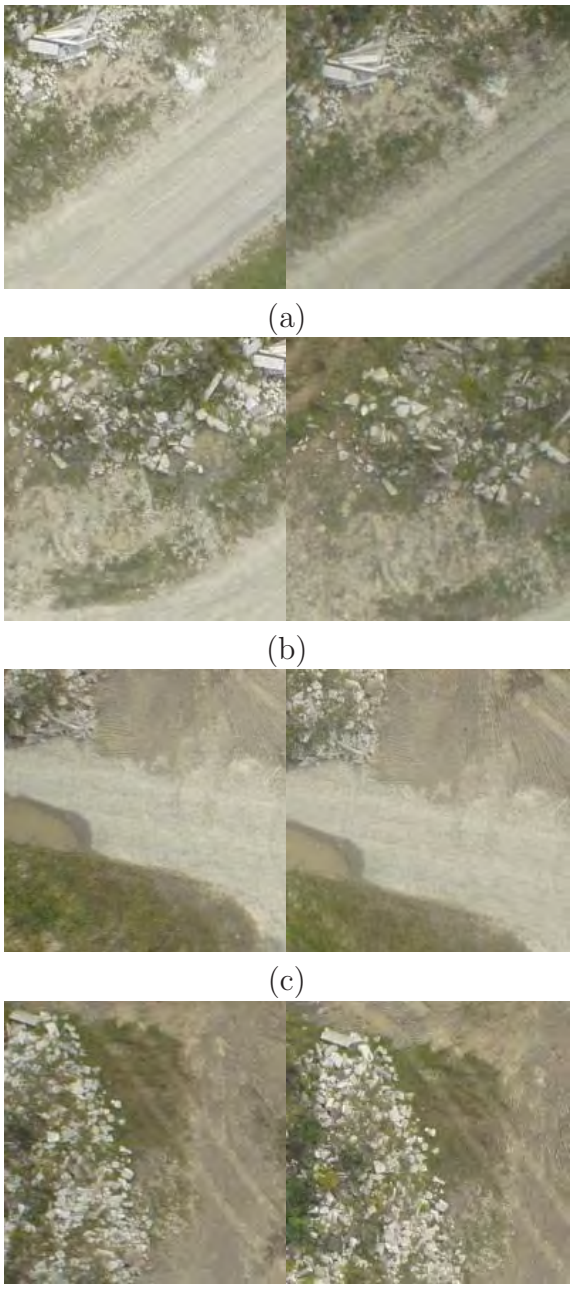
A.3 shows the learned layer-1 filters on rocks images when projected back into pixel space. It can be seen that the layer-1 filters learn Gabor-style filters. Compared with layer-1 filters learned from vehicles A.9, these filters contain little color information.

A.4 shows the learned layer-2 filters on rocks images when projected back into pixel space. It can be seen that the layer-2 filters are highly diverse but contain little color information compared with layer-2 filters learned from vehicles (A.10). However, many of these filters are various fine parallel lines and conjunctions of T-junctions.

A.5 and A.6 show the matching results of my proposed method and the standard way of using SIFT on rock images, respectively.



Figure A.1: Original aerial images of rocks



(a)

(b)

(c)

(d)

Figure A.2: Extracted patches from aerial images of rocks

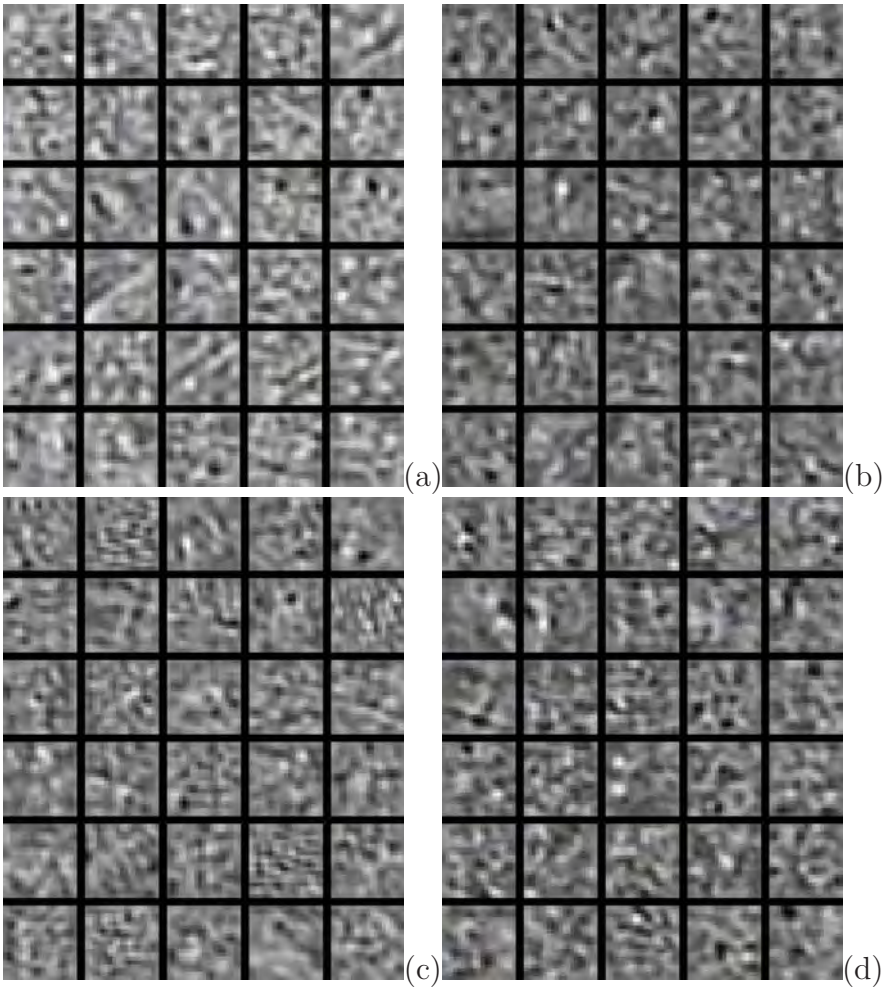


Figure A.3: Layer-1 filters on rocks images projected back into pixel space

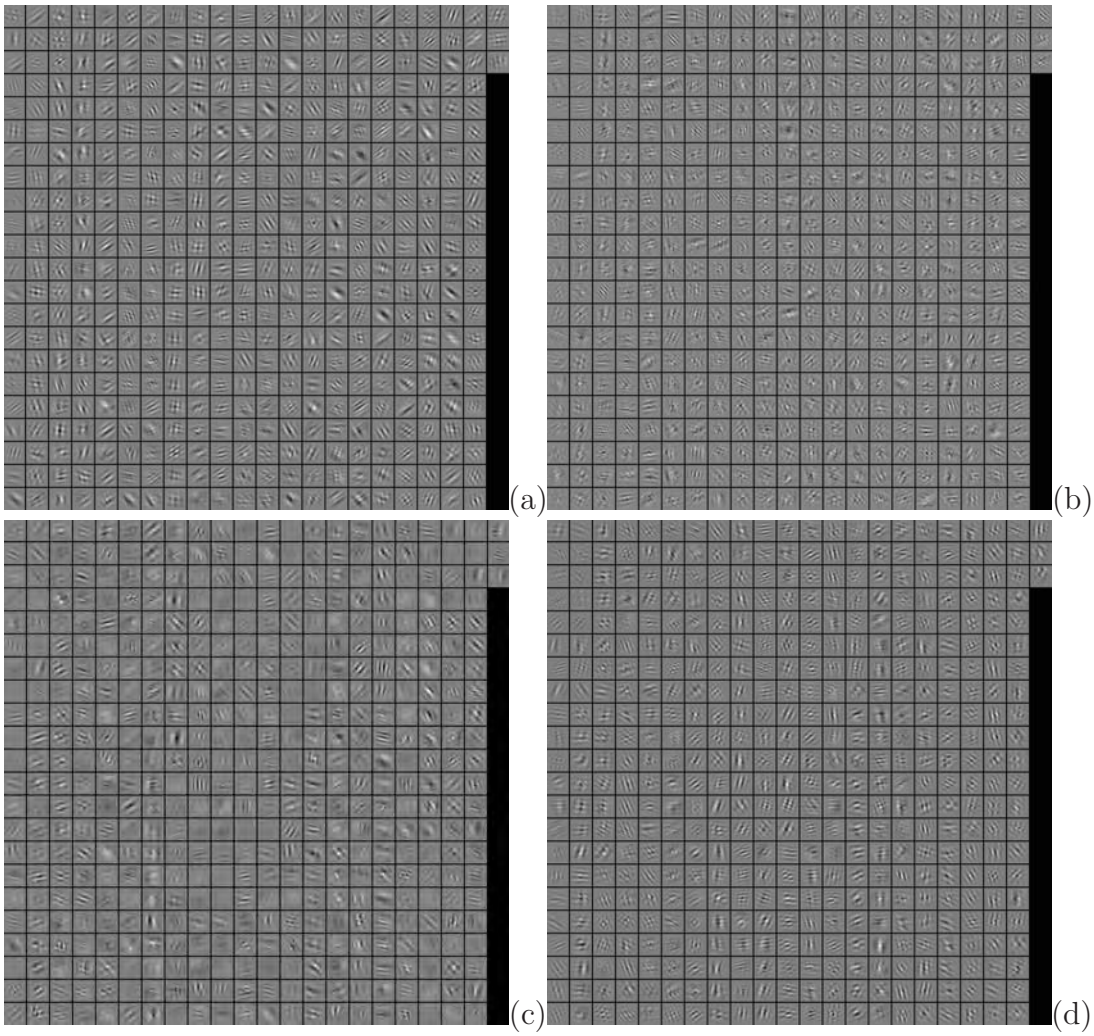
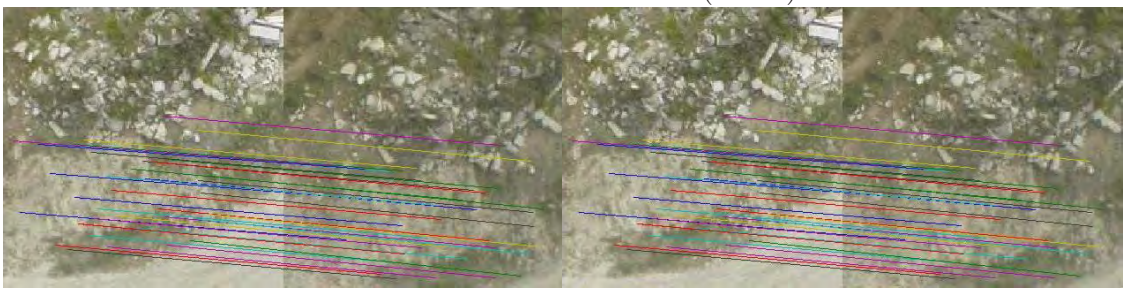


Figure A.4: Layer-2 filters on rocks images projected back into pixel space



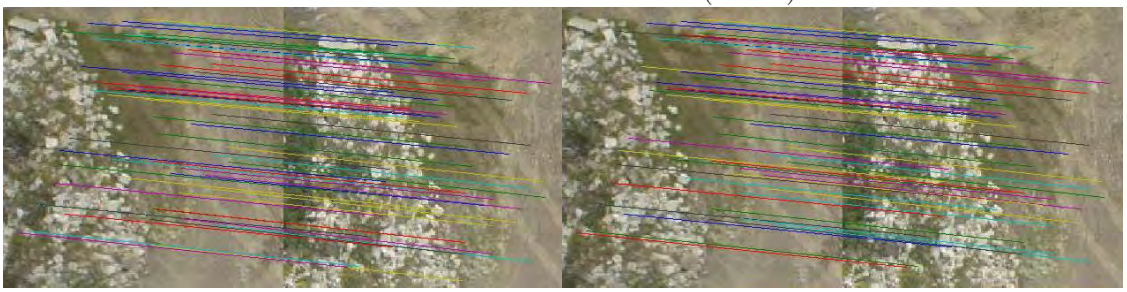
61 tentative matches | 61 (100%) inlier matches



40 tentative matches | 40 (100%) inlier matches



54 tentative matches | 54 (100%) inlier matches

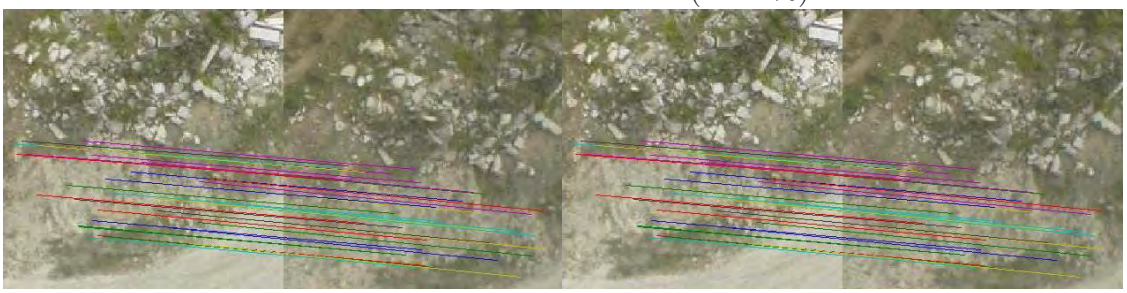


60 tentative matches | 58 (96.67%) inlier matches

Figure A.5: Matching results of my proposed method on rocks images



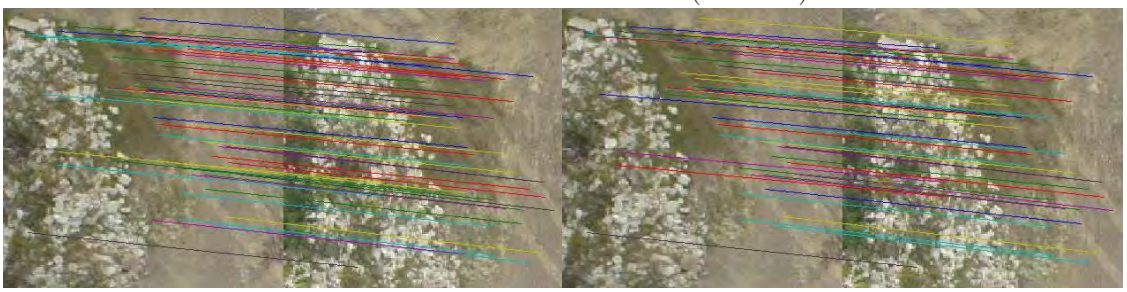
61 tentative matches | 58 (95.08%) inlier matches



40 tentative matches | 40 (100%) inlier matches



54 tentative matches | 50 (92.59%) inlier matches



60 tentative matches | 57 (95%) inlier matches

Figure A.6: Matching results of SIFT detector and SIFT descriptor on rocks images



Figure A.7: Original aerial images of vehicles

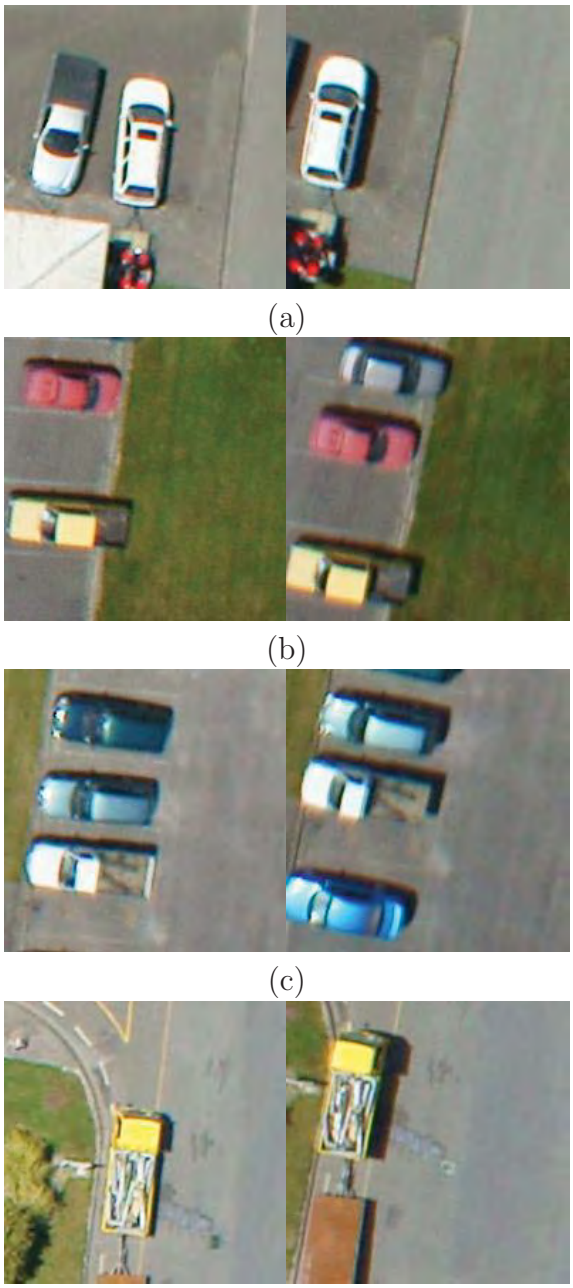
A.2 Vehicles images

A.7 shows the pair of original 1616 × 1080 RGB aerial images of vehicles.

A.8 shows the extracted 200 × 200 RGB patches from original aerial images of vehicles.

A.9 shows the learned layer-1 filters on vehicles image patches when projected back into pixel space. It can be seen that the layer-1 filters learn colorful Gabor-style filters.

A.10 shows the learned layer-2 filters on vehicles image patches when projected back into pixel space. It can be seen that the layer-2 filters are highly diverse and include rich color information. Since vehicles image patches have regular shapes, many of the layer-2 filters are oriented colorful parallel lines.



(a)

(b)

(c)

(d)

Figure A.8: Extracted patches from aerial images of vehicles

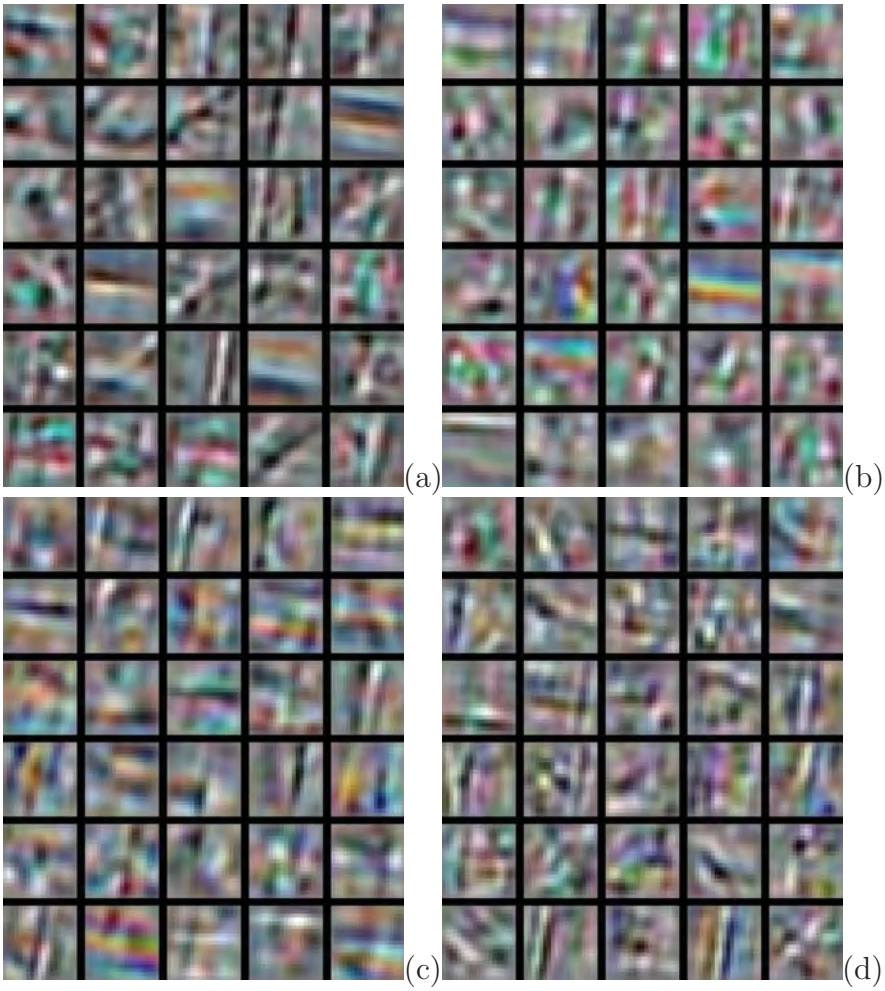


Figure A.9: Layer-1 filters of vehicles projected back into pixel space

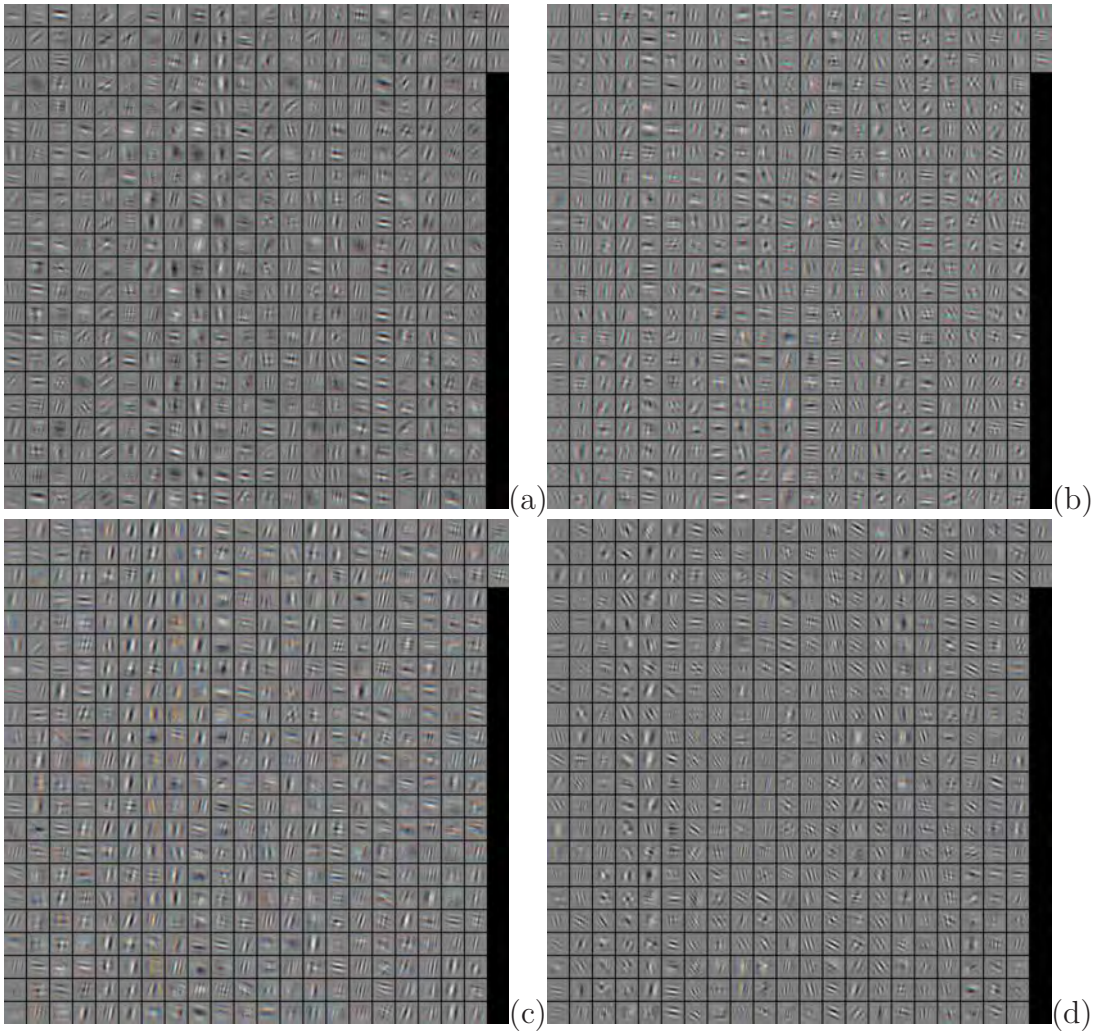


Figure A.10: layer-2 filters of vehicles projected back into pixel space



1 tentative matches | 1 (100%) inlier matches



12 tentative matches | 12 (100%) inlier matches



6 tentative matches | 6 (100%) inlier matches



5 tentative matches | 5 (100%) inlier matches

Figure A.11: Matching results of my proposed method on vehicles images



1 tentative matches | 1 (100%) inlier matches



12 tentative matches | 10 (83.33%) inlier matches



6 tentative matches | 6 (100%) inlier matches



5 tentative matches | 4 (80%) inlier matches

Figure A.12: Matching results of SIFT detector and SIFT descriptor on vehicles images



Figure A.13: Original aerial images of buildings (set 2)

A.3 Buildings

A.13 shows the pair of original 1616 × 1080 RGB aerial images of buildings.

A.14 shows the extracted 200 × 200 RGB patches from the original aerial images of buildings (set 2).

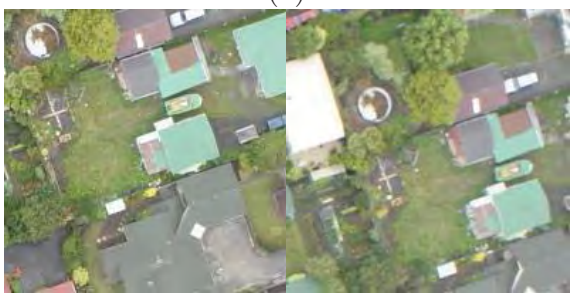
A.15 shows the learned layer-1 filters on buildings images (set 2) when projected into pixel space. It can be seen that the layer-1 filters learned colorful Gabor-style elements.

A.16 shows the learned layer-2 filters on buildings images (set 2) when projected back into pixel space. It can be seen that the layer-2 filters are highly diverse and include rich color information. Since buildings images have regular shapes, many of them are oriented colorful parallel lines and conjunctions of T-junctions.

A.17 and A.18 show the matching results of my proposed method and the standard way of using SIFT on buildings images (set 2), respectively.



(a)



(b)



(c)



(d)

Figure A.14: Extracted patches from aerial images of buildings (set 2)

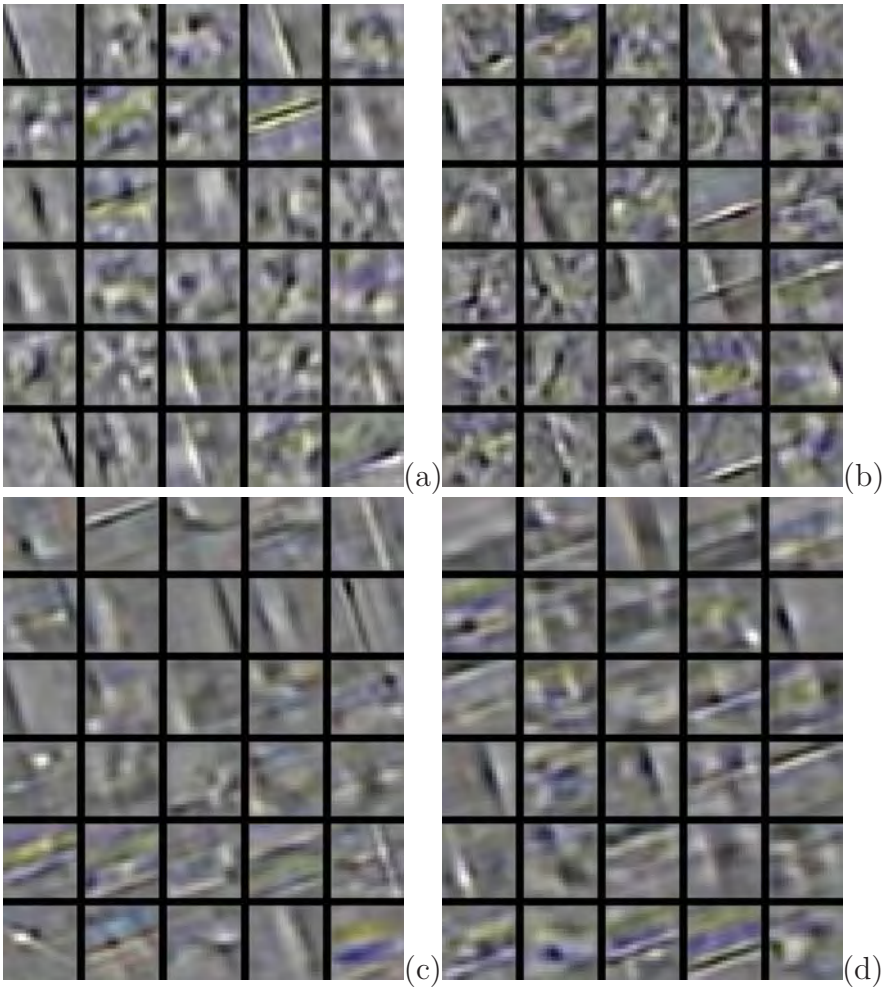


Figure A.15: Layer-1 filters on buildings images (set 2) projected back into pixel space

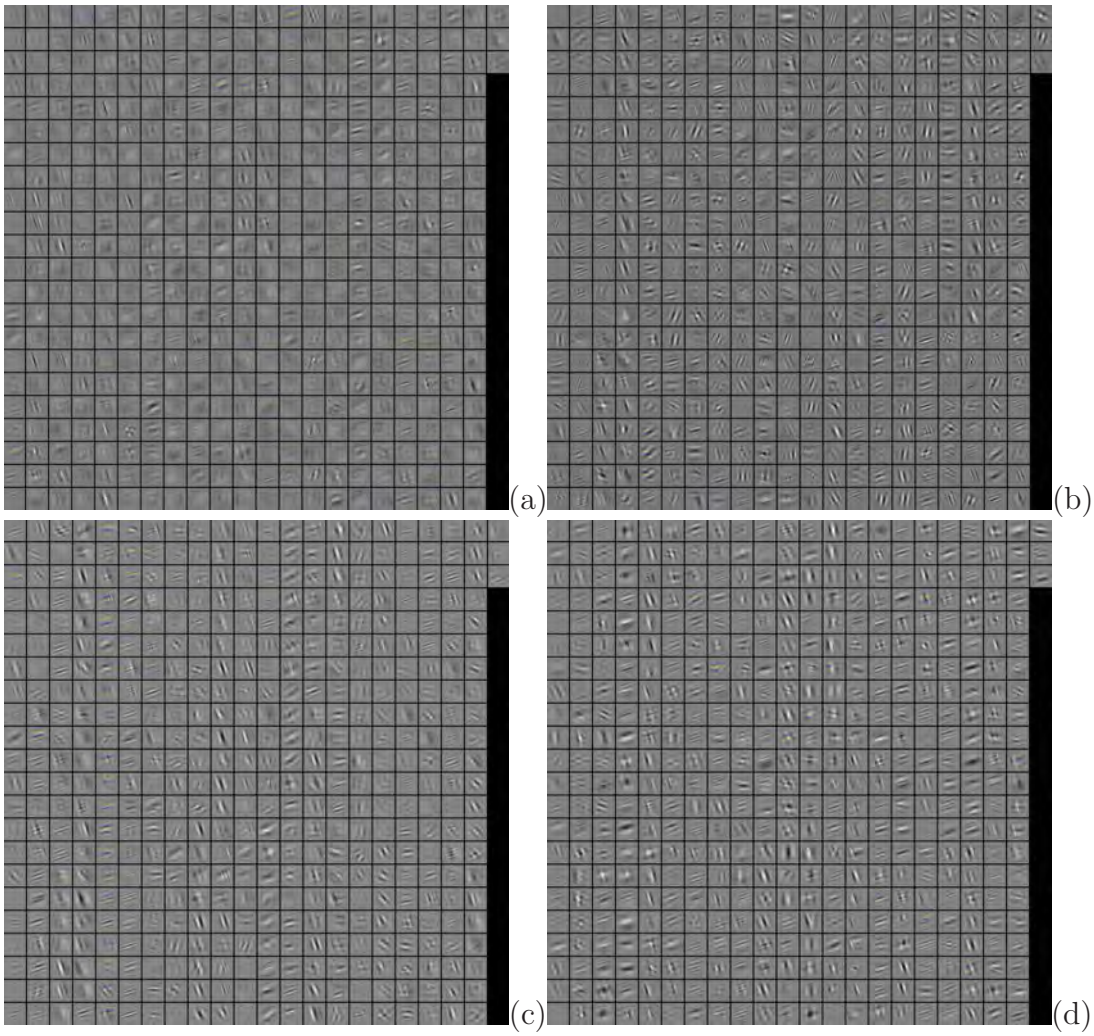


Figure A.16: Layer-2 filters on buildings images (set 2) projected back into pixel space



51 tentative matches 42 (82.35%) inlier matches



29 tentative matches 24 (82.76%) inlier matches



21 tentative matches 18 (85.71%) inlier matches



40 tentative matches 21 (52.5%) inlier matches

Figure A.17: Matching results of my proposed method on buildings images



51 tentative matches | 49 (96.08%) inlier matches



29 tentative matches | 27 (93.1%) inlier matches



21 tentative matches | 16 (76.19%) inlier matches



38 tentative matches | 18 (47.37%) inlier matches

Figure A.18: Matching results of SIFT detector and SIFT descriptor on buildings

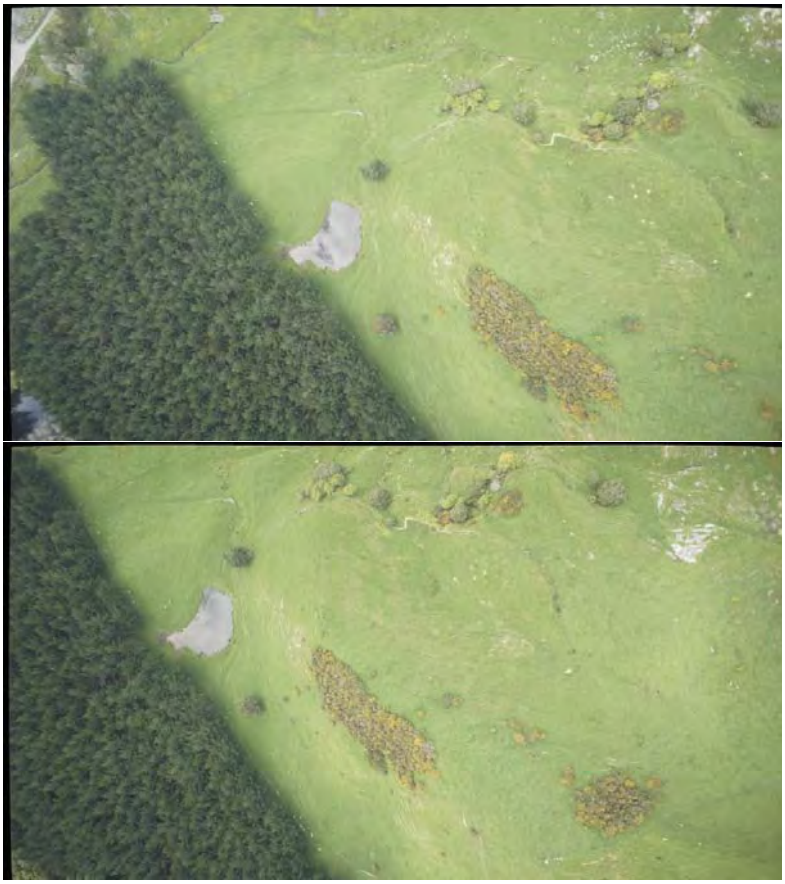


Figure A.19: Original aerial images of forest (set 1)

A.4 Forest images

A.4.1 Set 1

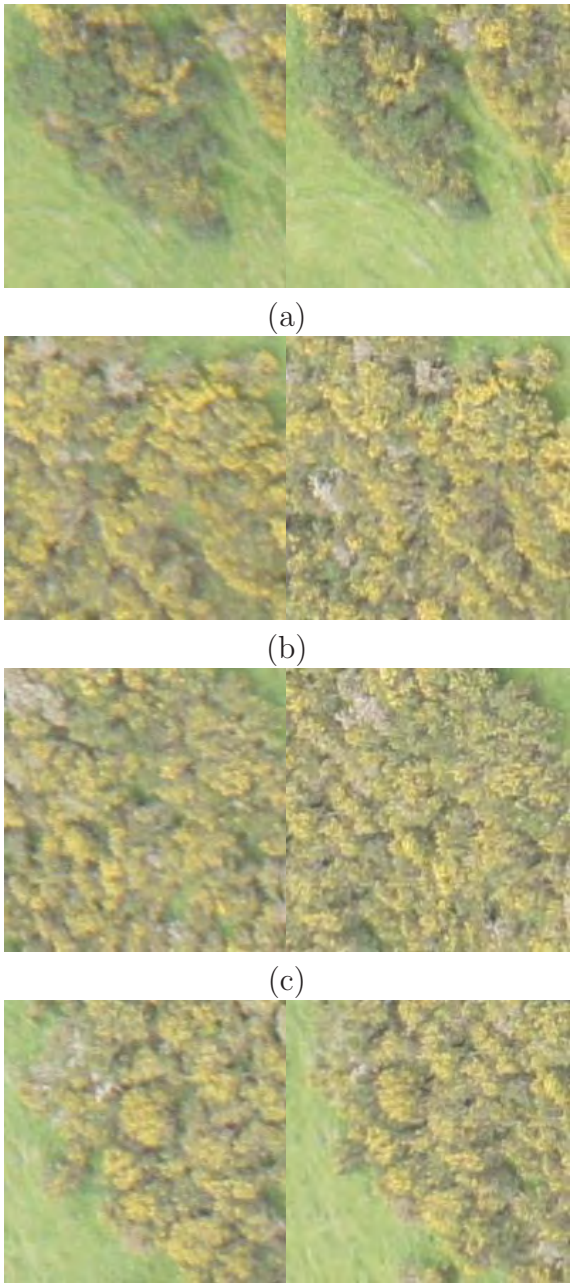
A.19 shows the pair of original 4592 × 2576 RGB aerial images of forest (set 1).

A.20 shows the extracted 200 × 200 RGB patches from original aerial images of forest (set 1).

A.21 shows the learned layer-1 filters on forest images (set 1) when projected into pixel space. It can be seen that unlike the situations in vehicles or buildings which have regular shapes, the layer-1 filters learn somewhat chaotic colorful elements.

A.22 shows the learned layer-2 filters on forest images (set 1) when projected back into pixel space. It can be seen that the layer-2 filters are highly diverse and include rich color information. Many of them are oriented colorful parallel lines and conjunction of T-junctions.

A.23 and A.24 show the matching results of my proposed method and the standard way of using SIFT on forest images (set 1), respectively.



(a)

(b)

(c)

(d)

Figure A.20: Extracted patches from aerial images of forest (set 1)

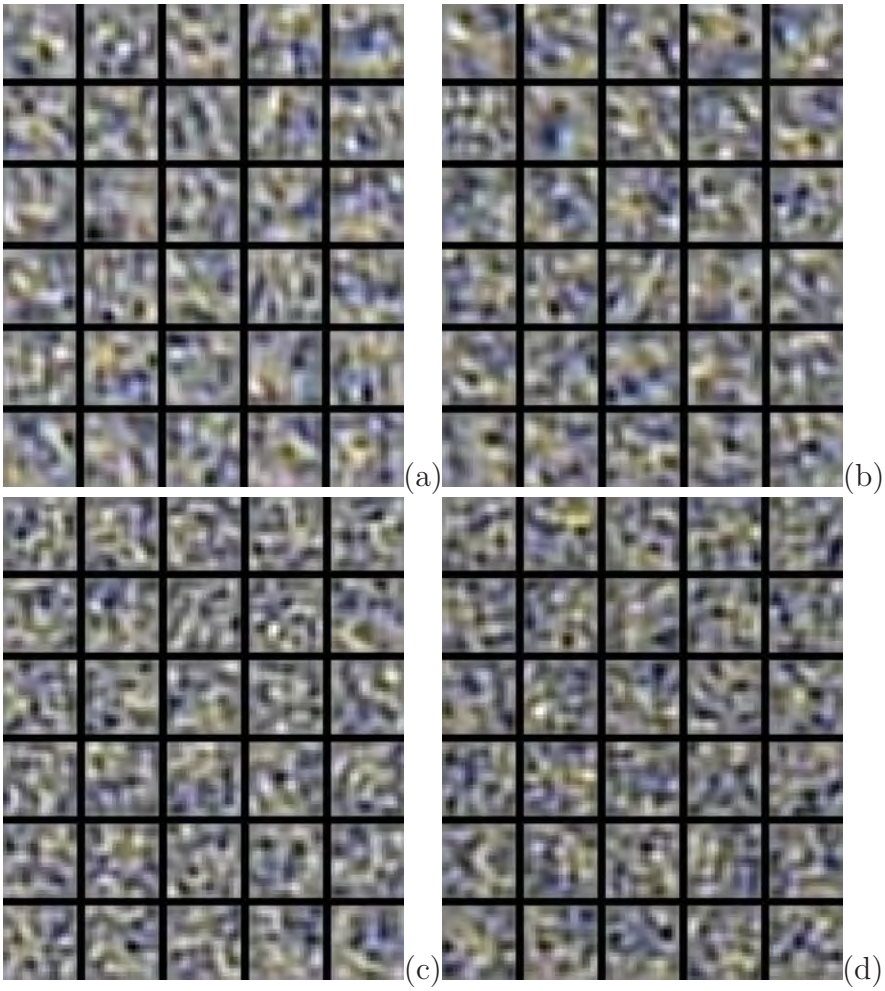


Figure A.21: Layer-1 filters on forest images (set 1) projected back into pixel space

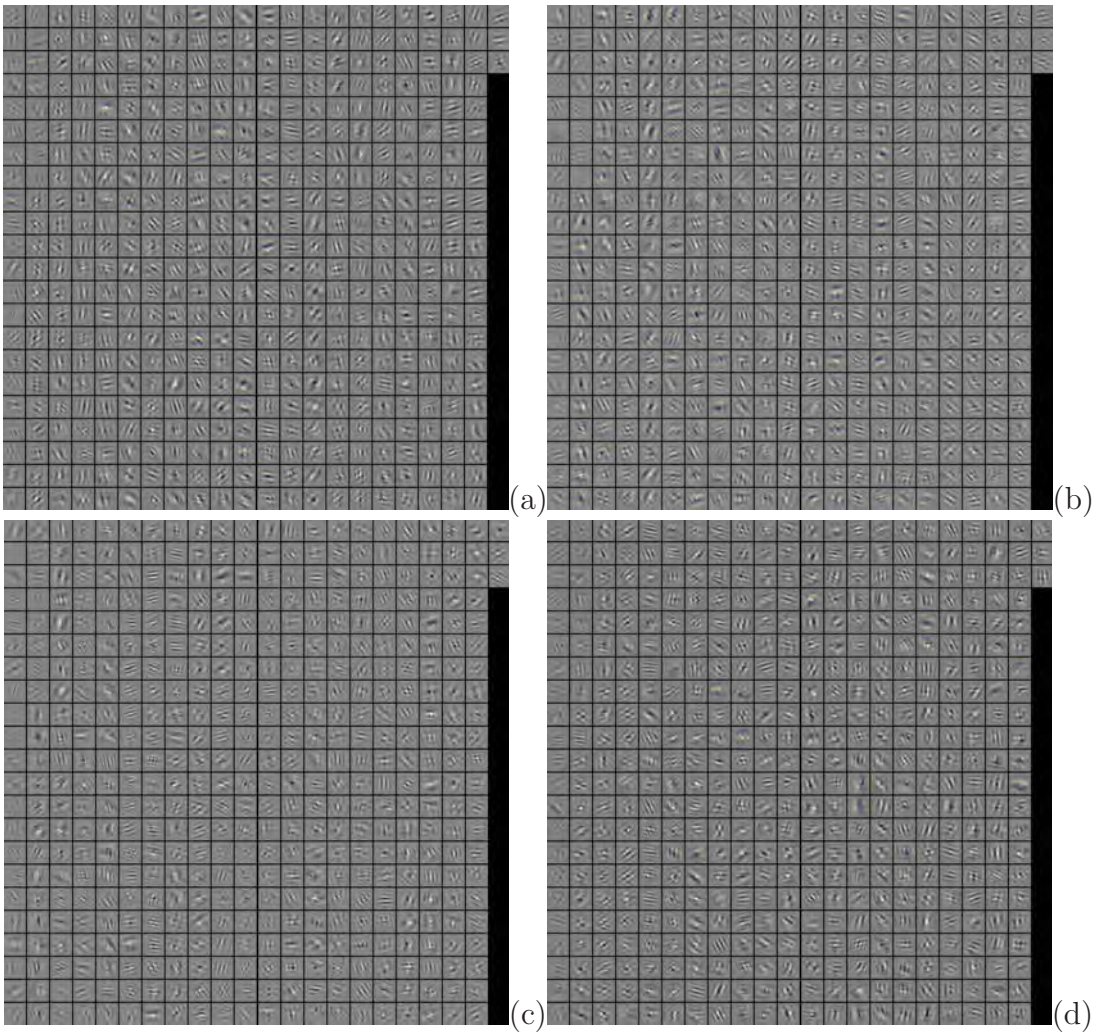


Figure A.22: Layer-2 filters on forest images (set 1) projected back into pixel space

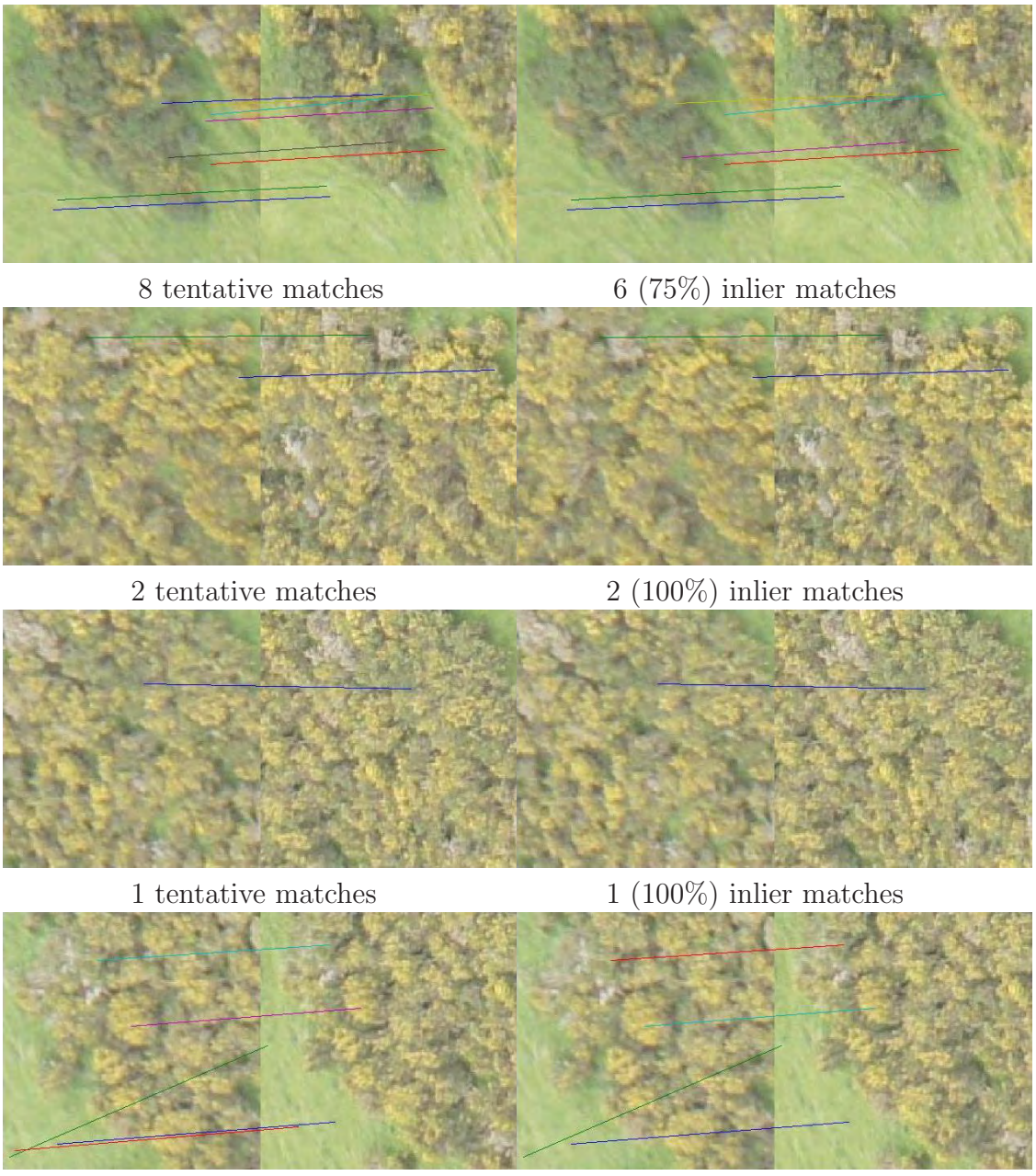
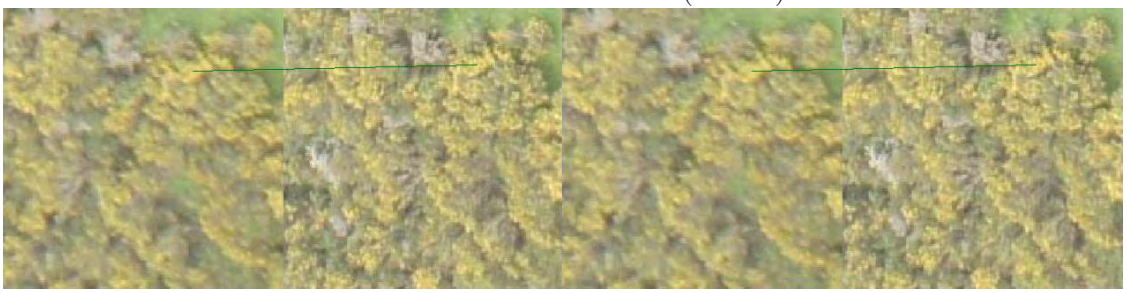


Figure A.23: Matching results of my proposed method on forest images (set 1)



8 tentative matches | 7 (87.5%) inlier matches



2 tentative matches | 2 (100%) inlier matches



1 tentative matches | 1 (100%) inlier matches



5 tentative matches | 5 (100%) inlier matches

Figure A.24: Matching results of SIFT detector and SIFT descriptor on forest images (set 1)



Figure A.25: Original aerial images of forest (set 2)

A.4.2 Set 2

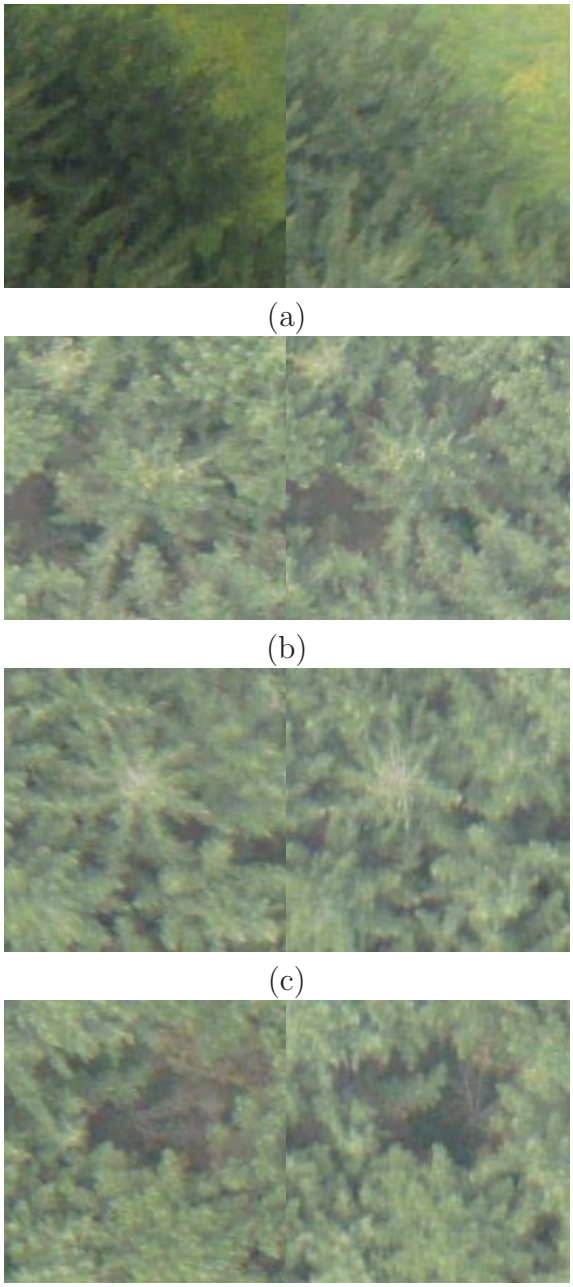
A.25 shows the pair of original 4592 × 2576 RGB aerial images of forest (set 2).

A.26 shows the extracted 200 × 200 RGB patches from original aerial images of forest (set 2).

A.27 shows the learned layer-1 filters on forest images (set 2) when projected into pixel space. It can be seen that unlike the situations in vehicles or buildings which have regular shapes, the layer-1 filters learn somewhat chaotic colorful elements.

A.28 shows the learned layer-2 filters on forest images (set 2) when projected back into pixel space. It can be seen that the layer-2 filters are highly diverse but include little color information. Many of them are oriented parallel lines and conjunction of T-junctions.

A.29 and A.30 show the matching results of my proposed method and the standard way of using SIFT on forest images (set 2), respectively. It should be noted that although the percentage of inlier matches of SIFT method is around 50%, the actual matching points are always wrong. By contrast, my proposed method can always give the right matching points. However, for the fist pair of images, my proposed method gives no result.



(a)

(b)

(c)

(d)

Figure A.26: Extracted patches from aerial images of forest (set 2)

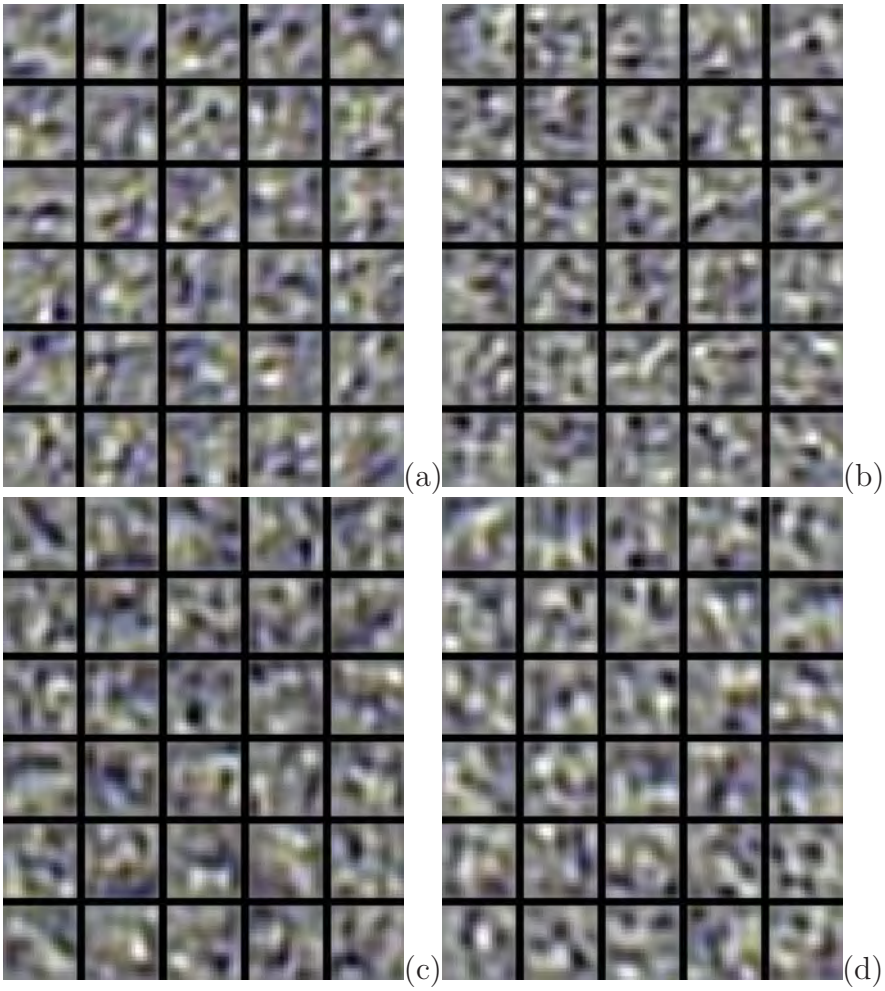


Figure A.27: Layer-1 filters on forest images (set 2) projected back into pixel space

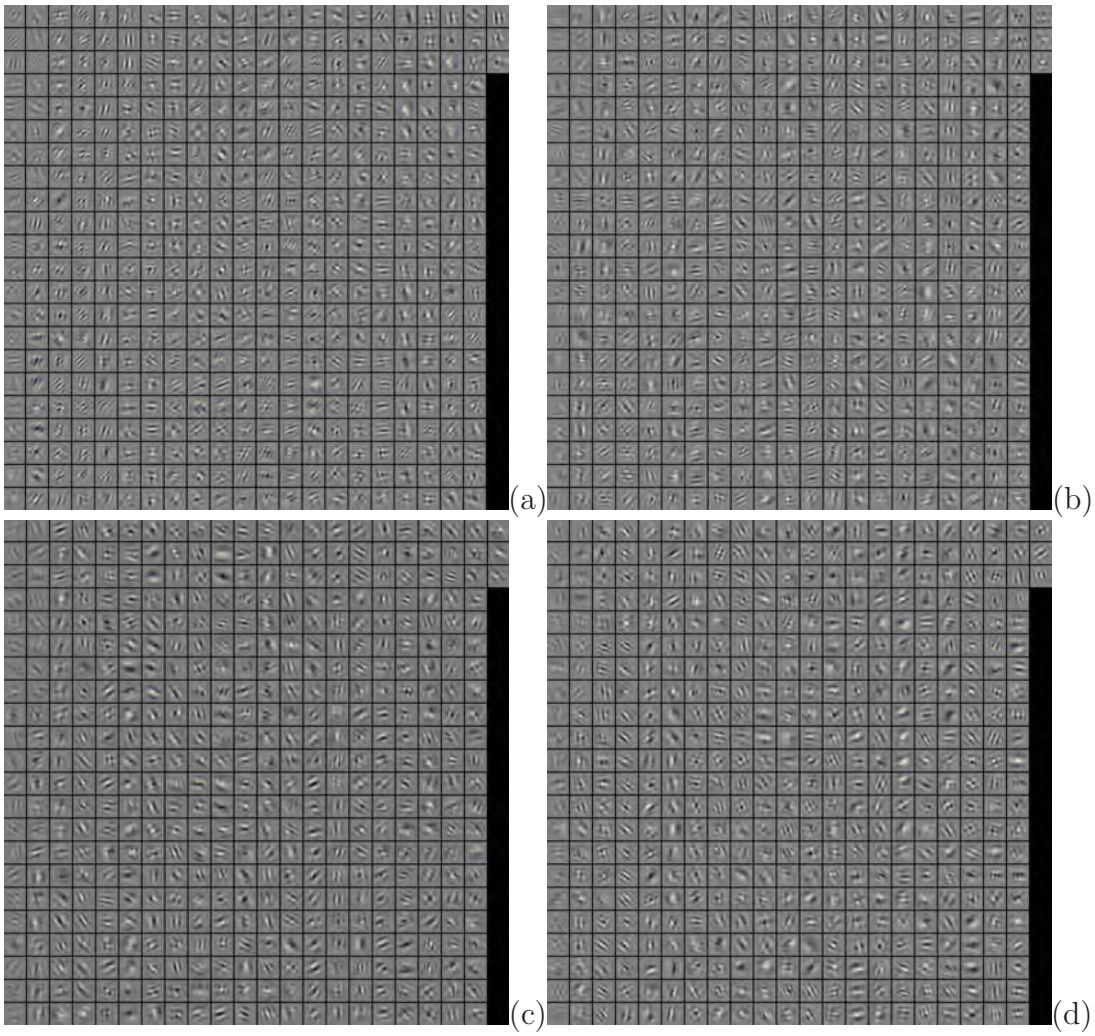


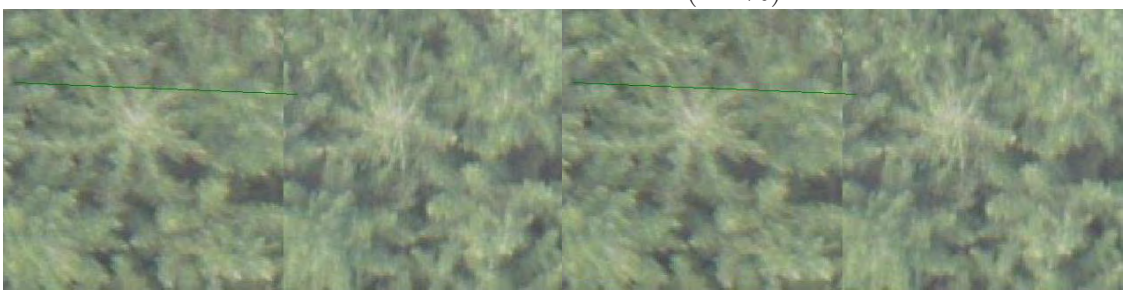
Figure A.28: Layer-2 filters on forest images (set 2) projected back into pixel space



1 tentative matches 1 (100%) inlier matches



1 tentative matches 1 (100%) inlier matches



2 tentative matches 2 (100%) inlier matches



2 tentative matches 2 (100%) inlier matches

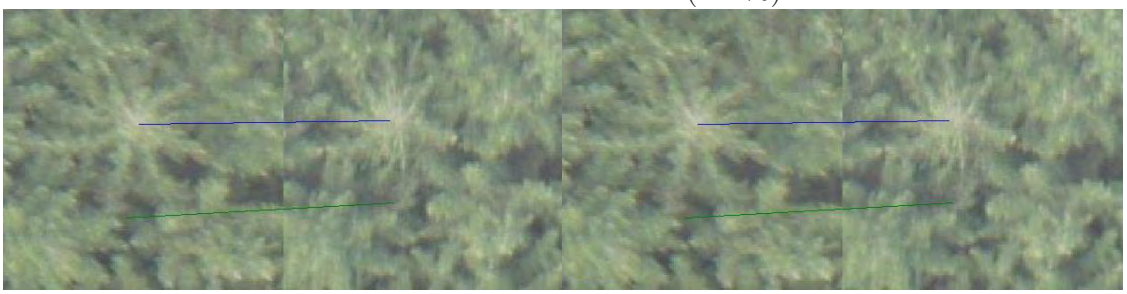
Figure A.29: Matching results of my proposed method on forest images (set 2)



1 tentative matches | 1 (100%) inlier matches



1 tentative matches | 1 (100%) inlier matches



2 tentative matches | 2 (100%) inlier matches



2 tentative matches | 2(100%) inlier matches

Figure A.30: Matching results of SIFT detector and SIFT descriptor on forest images (set 2)



Figure A.31: Original aerial images of forest (set 3)

A.4.3 Set 3

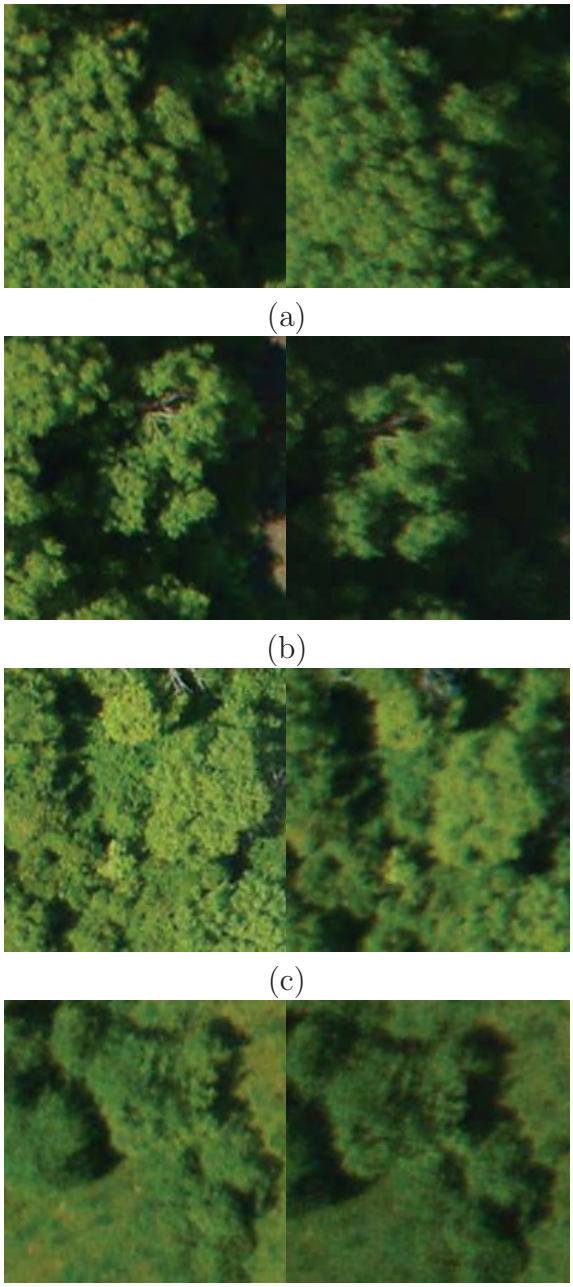
A.31 shows the pair of original 4592×2576 RGB aerial images of forest (set 3).

A.32 shows the extracted 200×200 RGB patches from original aerial images of forest (set 3).

A.33 shows the learned layer-1 filters on forest images (set 3) when projected into pixel space. It can be seen that unlike the situations in vehicles or buildings which have regular shapes, the layer-1 filters learn somewhat chaotic colorful elements.

A.34 shows the learned layer-2 filters on forest images (set 3) when projected back into pixel space. It can be seen that the layer-2 filters are highly diverse but include little color information. Many of them are oriented parallel lines and conjunction of T-junctions.

A.35 and A.36 show the matching results of my proposed method and the standard way of using SIFT on forest images (set 3), respectively.



(a)

(b)

(c)

(d)

Figure A.32: Extracted patches from aerial images of forest (set 3)

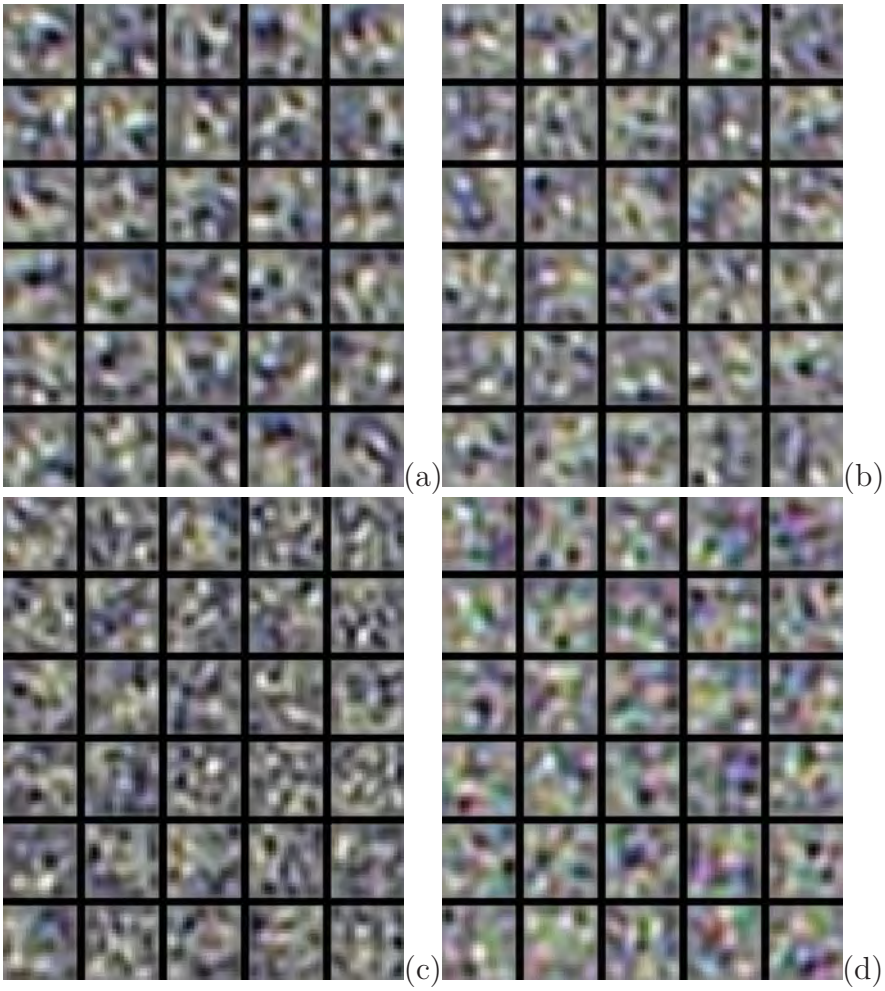


Figure A.33: Layer-1 filters on forest images (set 3) projected back into pixel space

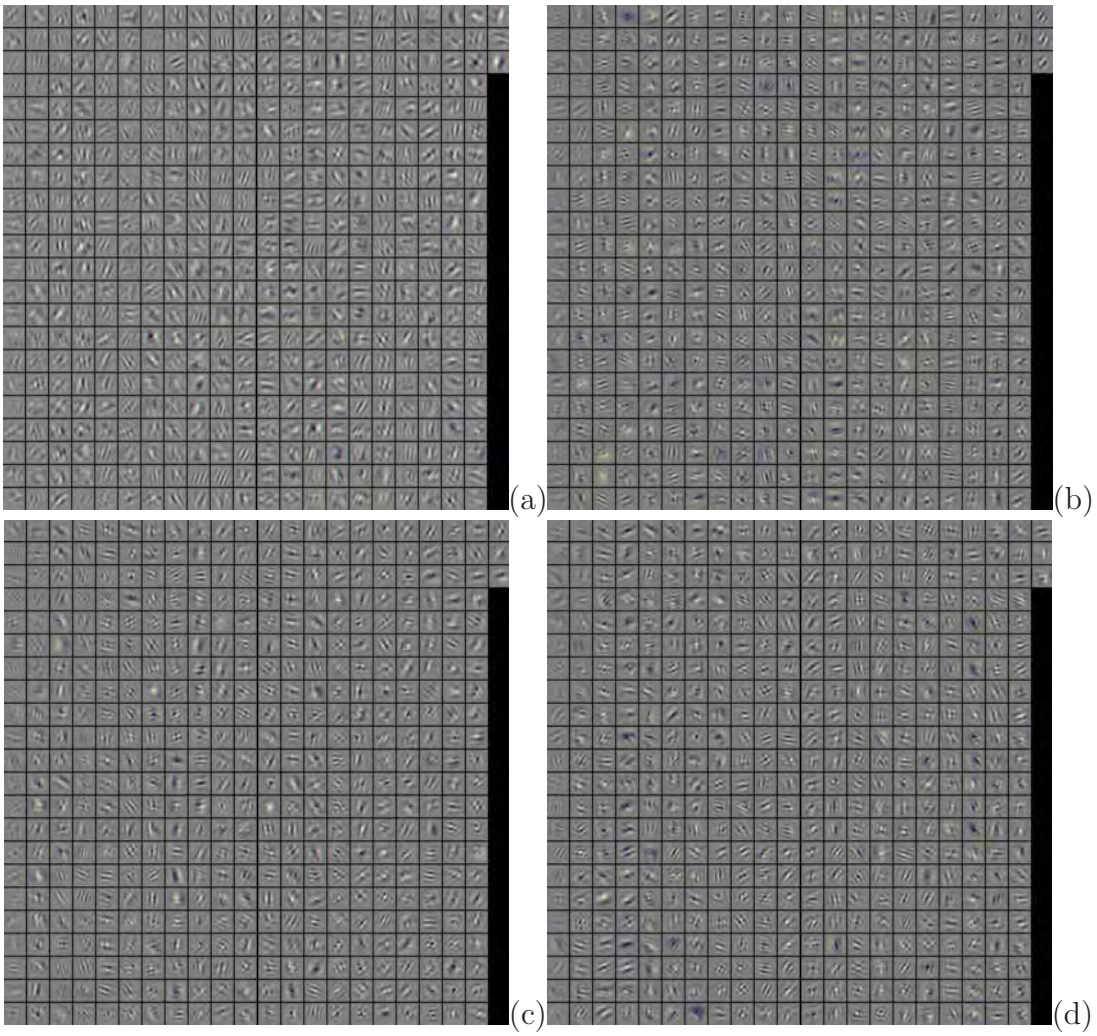
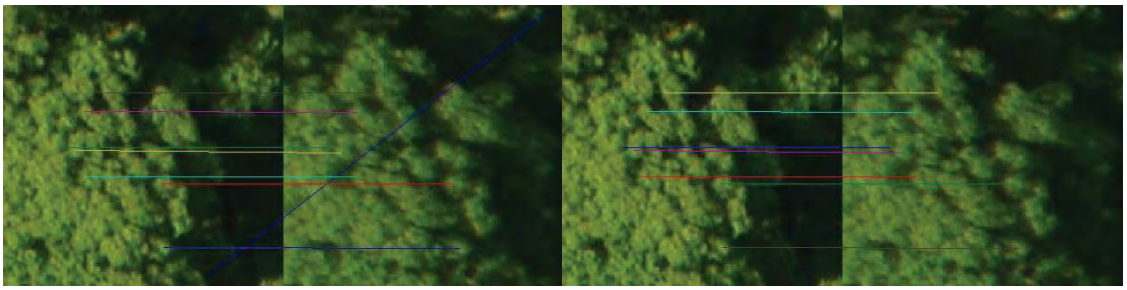
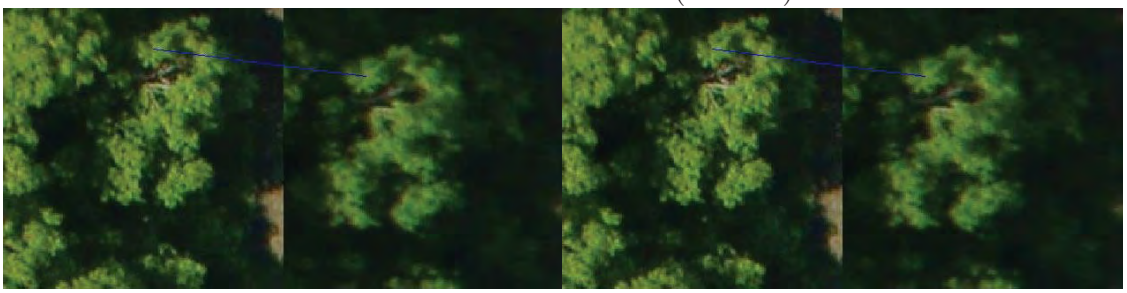


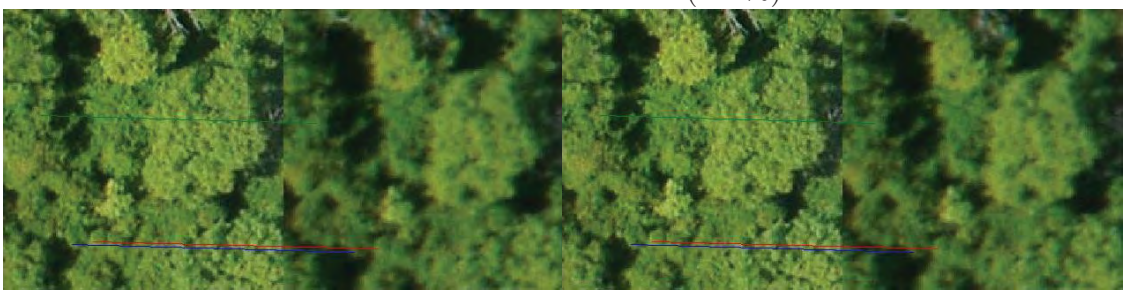
Figure A.34: Layer-2 filters on forest images (set 3) projected back into pixel space



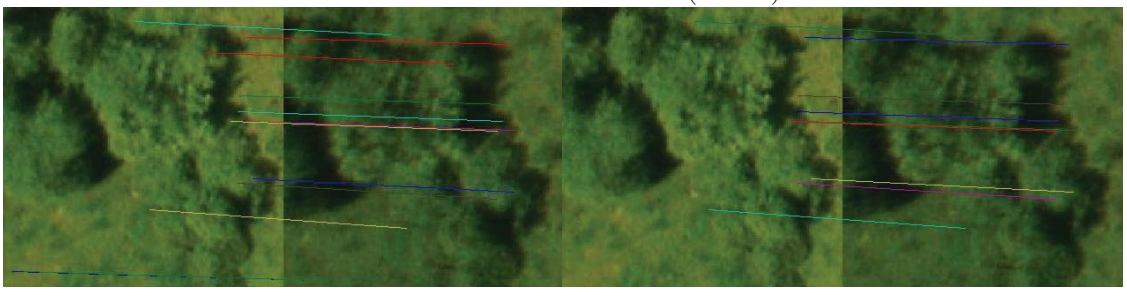
9 tentative matches | 8 (86.89%) inlier matches



1 tentative matches | 1 (100%) inlier matches

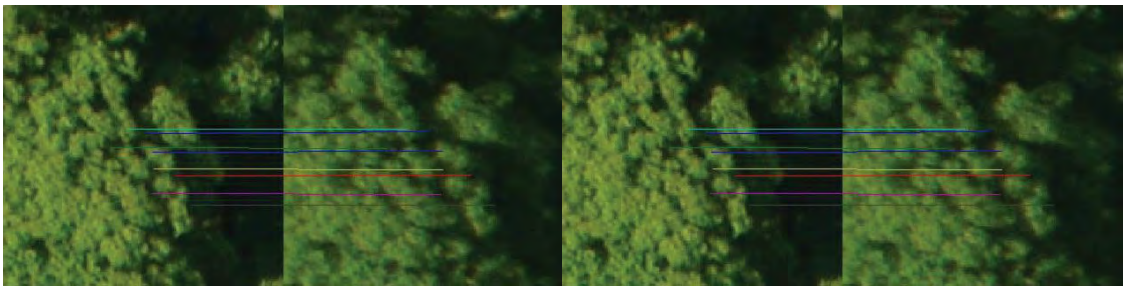


3 tentative matches | 3 (100%) inlier matches

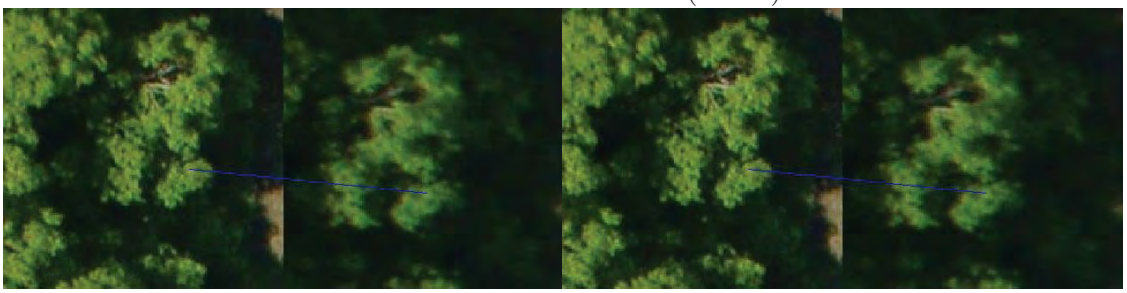


13 tentative matches | 10 (76.92%) inlier matches

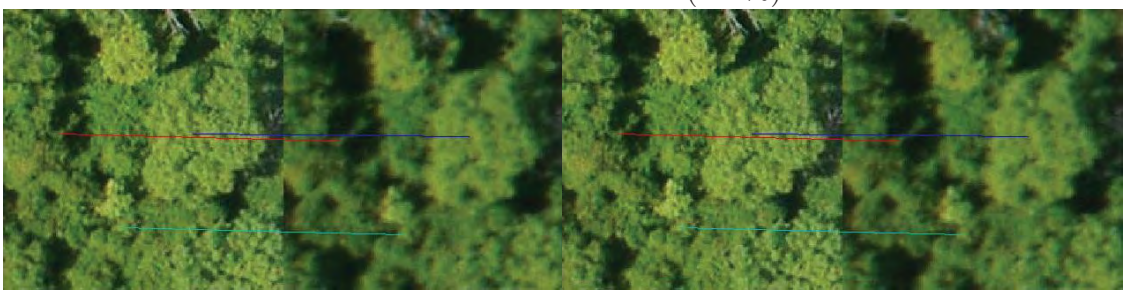
Figure A.35: Matching results of my proposed method on forest images (set 3)



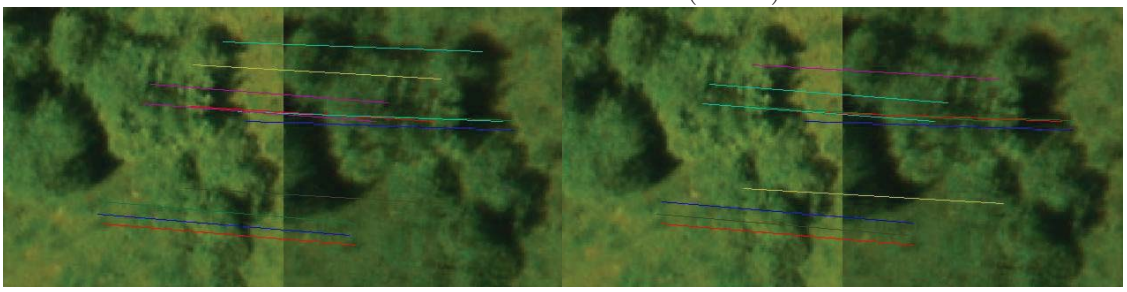
8 tentative matches | 8 (100%) inlier matches



1 tentative matches | 1 (100%) inlier matches



4 tentative matches | 4 (100%) inlier matches



13 tentative matches | 12 (92.31%) inlier matches

Figure A.36: Matching results of SIFT detector and SIFT descriptor on forest images (set 3)



Figure A.37: Original aerial images of forest (set 4)

A.4.4 Set 4

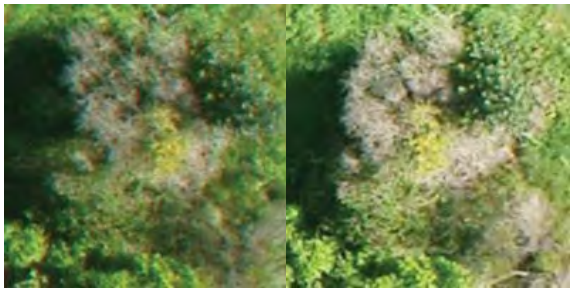
A.37 shows the pair of original 4592 × 2576 RGB aerial images of forest (set 4).

A.38 shows the extracted 200 × 200 RGB patches from original aerial images of forest (set 4).

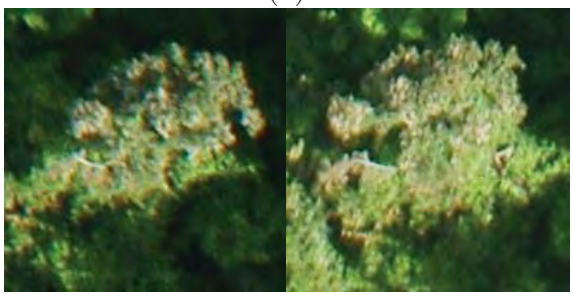
A.39 shows the learned layer-1 filters on forest images (set 4) when projected into pixel space. It can be seen that unlike the situations in vehicles or buildings which have regular shapes, the layer-1 filters learn somewhat chaotic colorful elements.

A.40 shows the learned layer-2 filters on forest images (set 4) when projected back into pixel space. It can be seen that the layer-2 filters are highly diverse and include rich color information. Many of them are oriented parallel lines and conjunction of T-junctions.

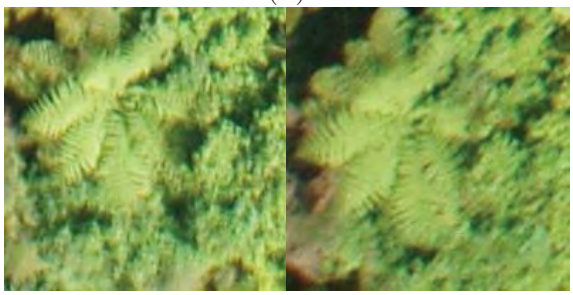
A.41 and A.42 show the matching results of my proposed method and the standard way of using SIFT on forest images (set 4), respectively.



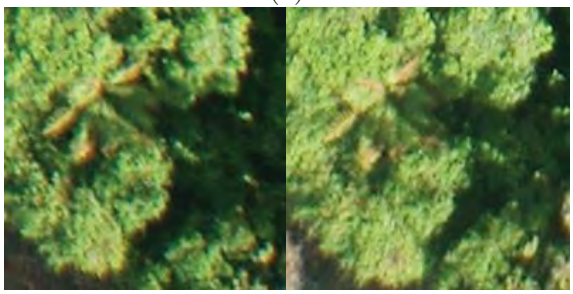
(a)



(b)



(c)



(d)

Figure A.38: Extracted patches from aerial images of forest (set 4)

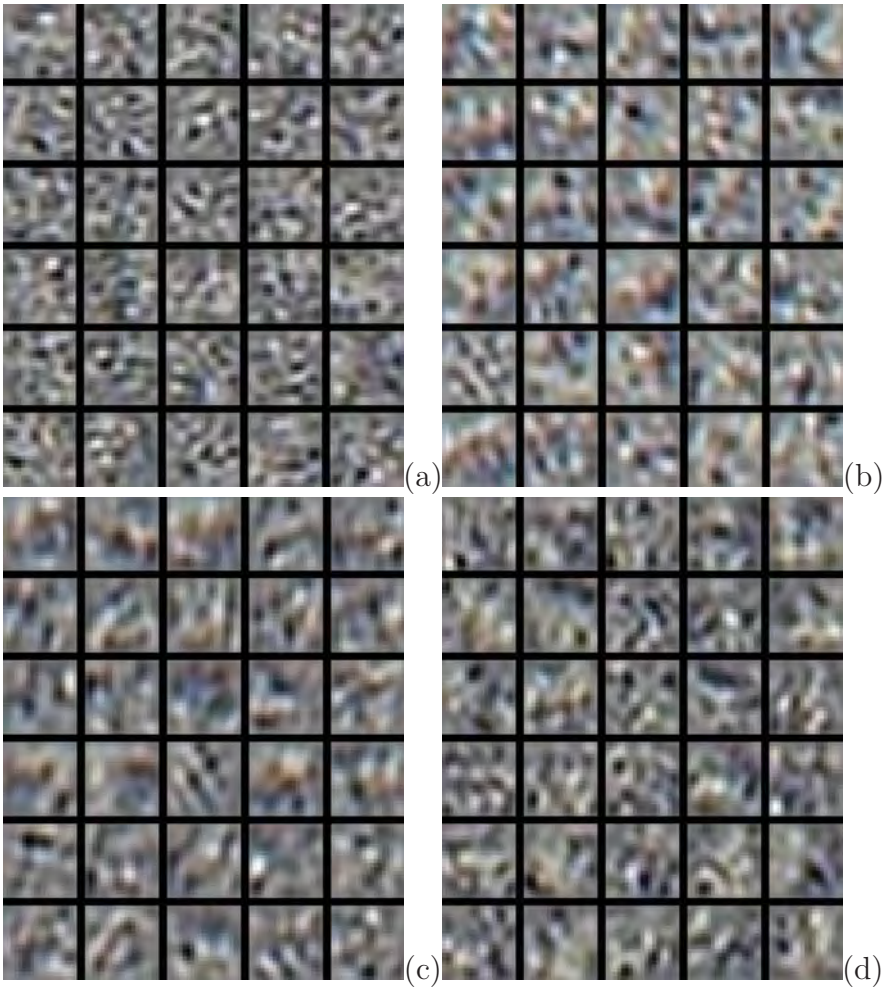


Figure A.39: Layer-1 filters on forest images (set 4) projected back into pixel space

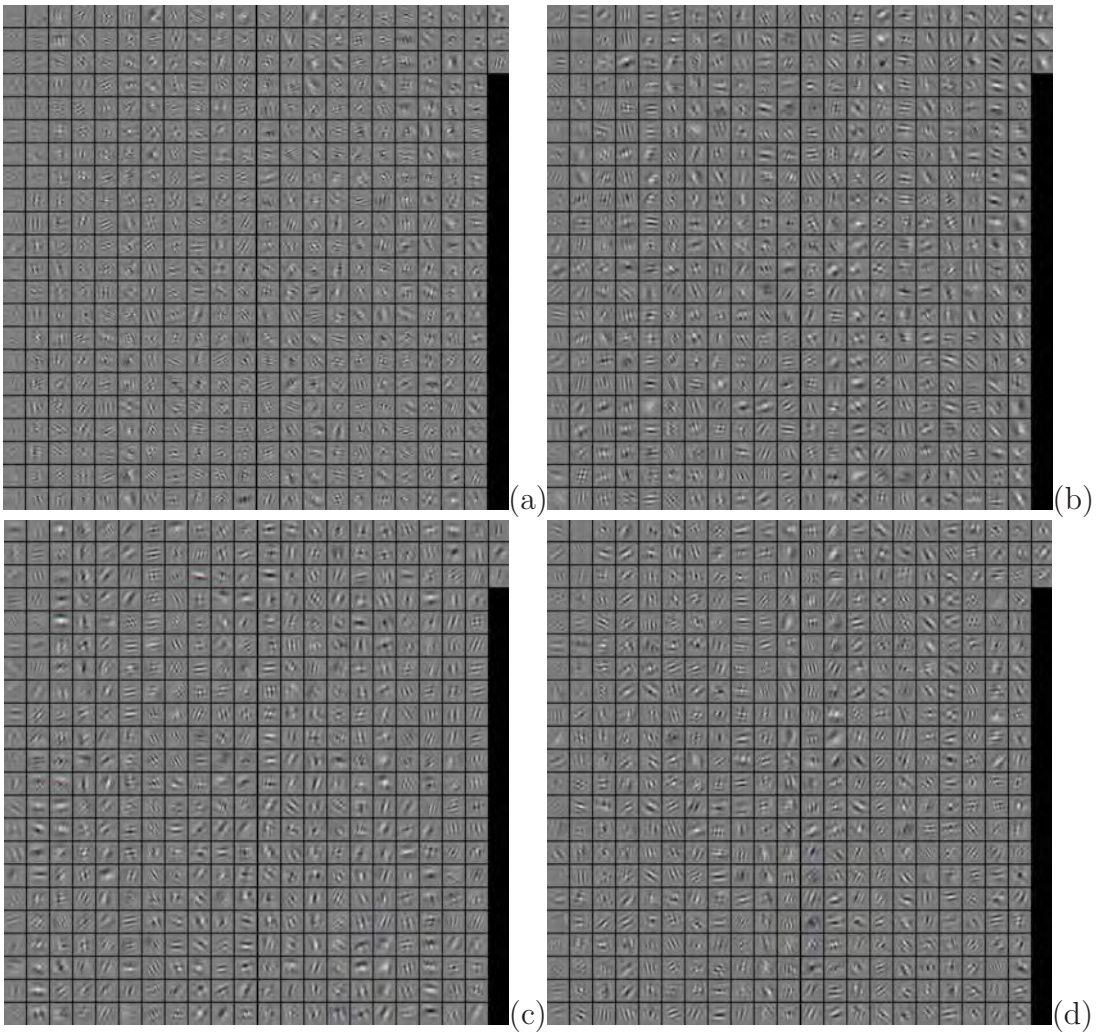
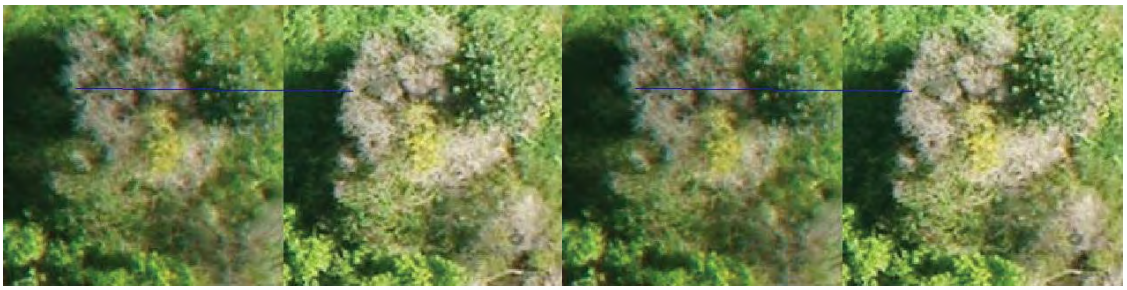
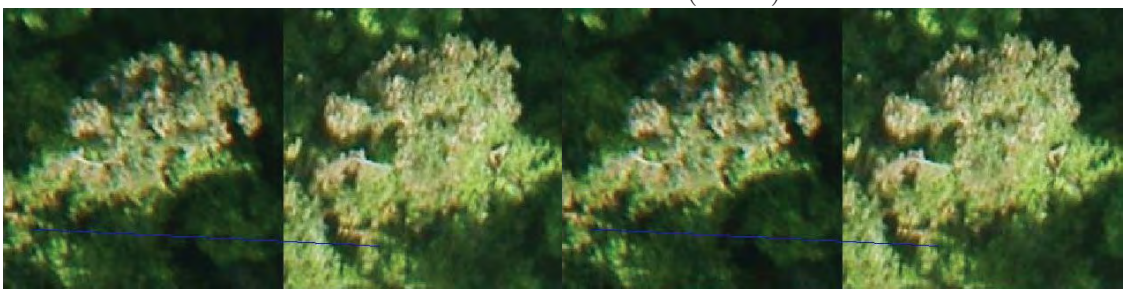


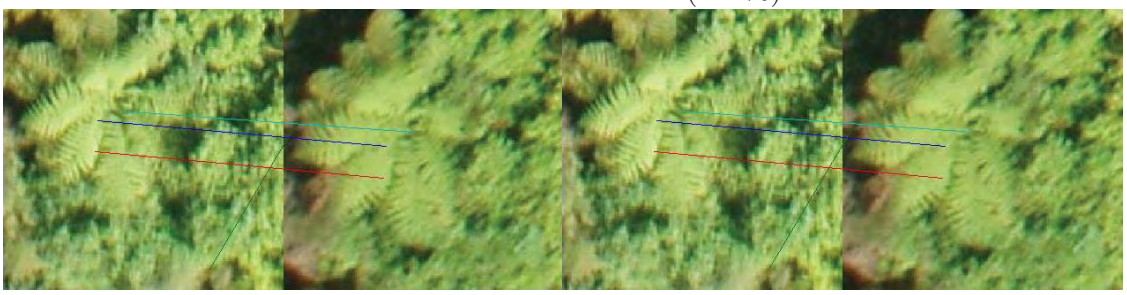
Figure A.40: Layer-2 filters on forest images (set 4) projected back into pixel space



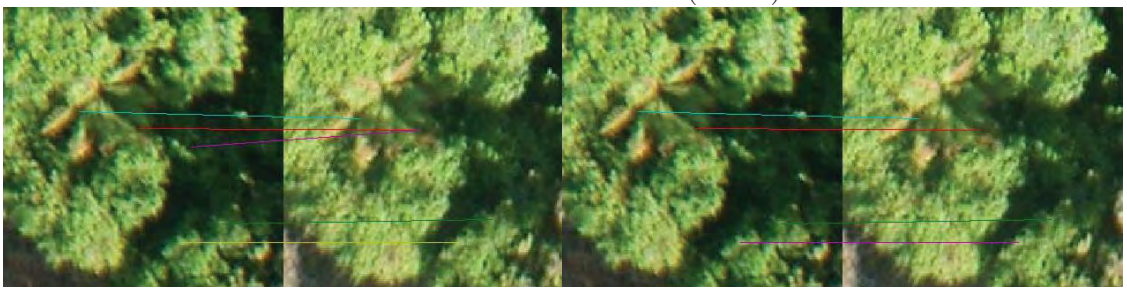
1 tentative matches | 1 (100%) inlier matches



1 tentative matches | 1 (100%) inlier matches

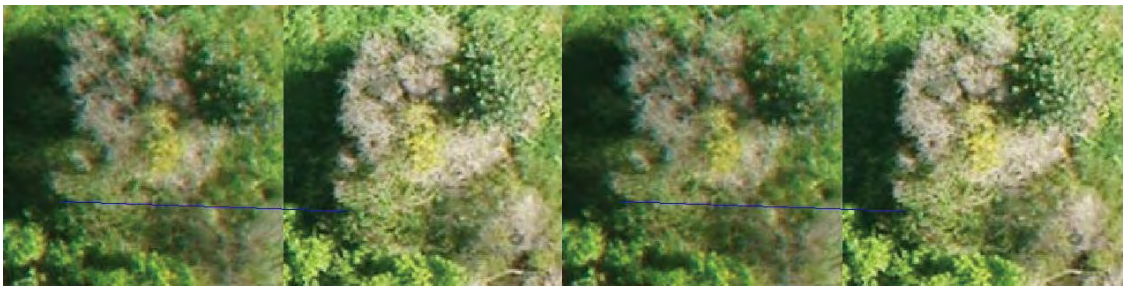


4 tentative matches | 4 (100%) inlier matches

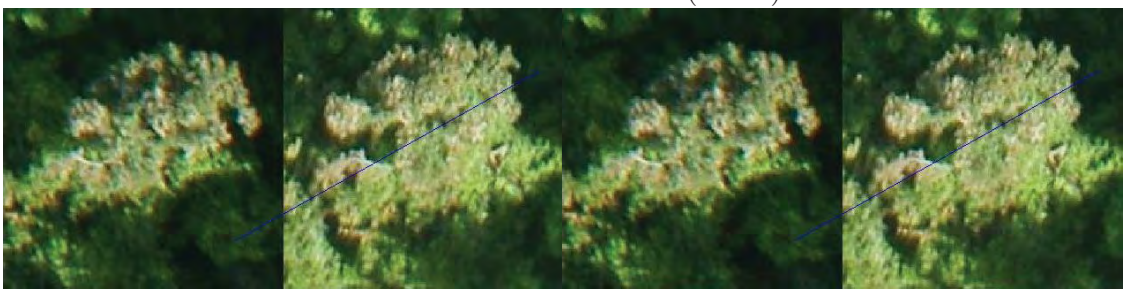


6 tentative matches | 5 (83.33%) inlier matches

Figure A.41: Matching results of my proposed method on forest images (set 4)



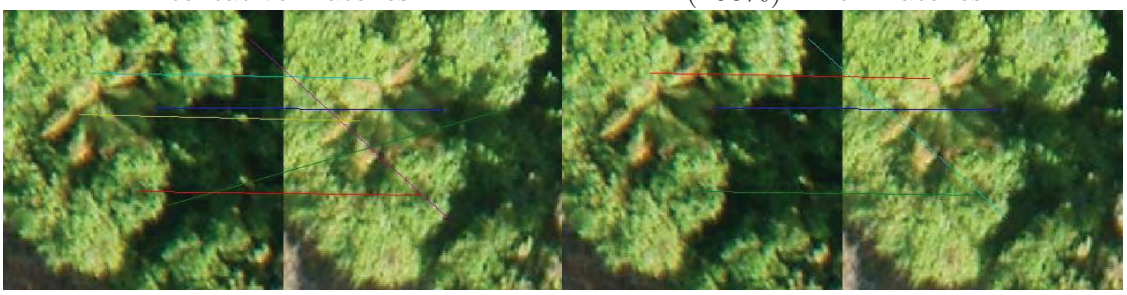
1 tentative matches | 1 (100%) inlier matches



1 tentative matches | 1 (100%) inlier matches



4 tentative matches | 4 (100%) inlier matches



6 tentative matches | 4 (66.67%) inlier matches

Figure A.42: Matching results of SIFT detector and SIFT descriptor on forest images (set 4)



Figure A.43: Original aerial images of forest (set 5)

A.4.5 Set 5

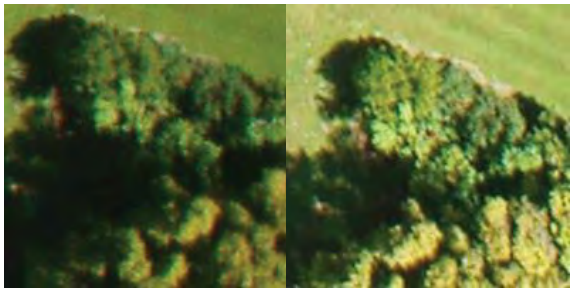
A.43 shows the pair of original 4592 × 2576 RGB aerial images of forest (set 5).

A.44 shows the extracted 200 × 200 RGB patches from original aerial images of forest (set 5).

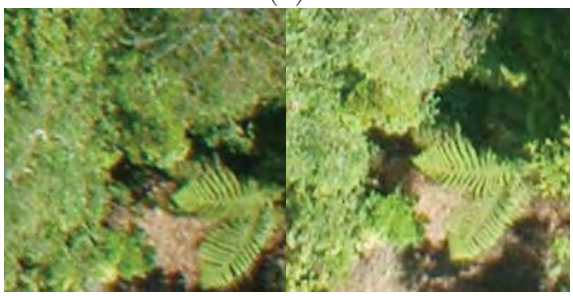
A.45 shows the learned layer-1 filters on forest images (set 5) when projected into pixel space. It can be seen that unlike the situations in vehicles or buildings which have regular shapes, the layer-1 filters learn somewhat chaotic colorful elements.

A.46 shows the learned layer-2 filters on forest images (set 5) when projected back into pixel space. It can be seen that the layer-2 filters are highly diverse but include little color information. Many of them are oriented parallel lines and conjunction of T-junctions.

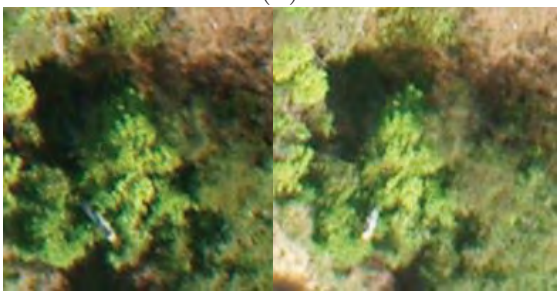
A.47 and A.48 show the matching results of my proposed method and the standard way of using SIFT on forest images (set 5), respectively.



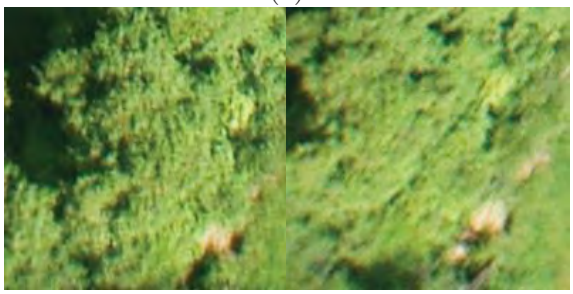
(a)



(b)



(c)



(d)

Figure A.44: Extracted patches from aerial images of forest (set 5)

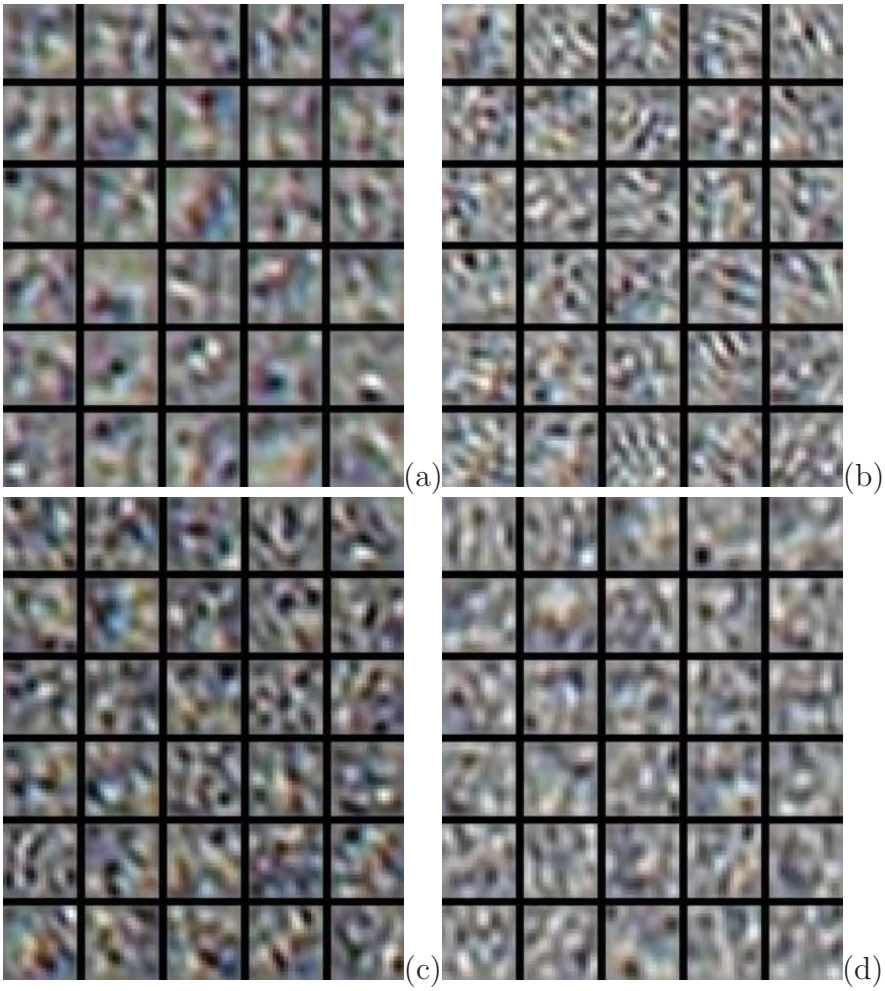


Figure A.45: Layer-1 filters on forest images (set 5) projected back into pixel space

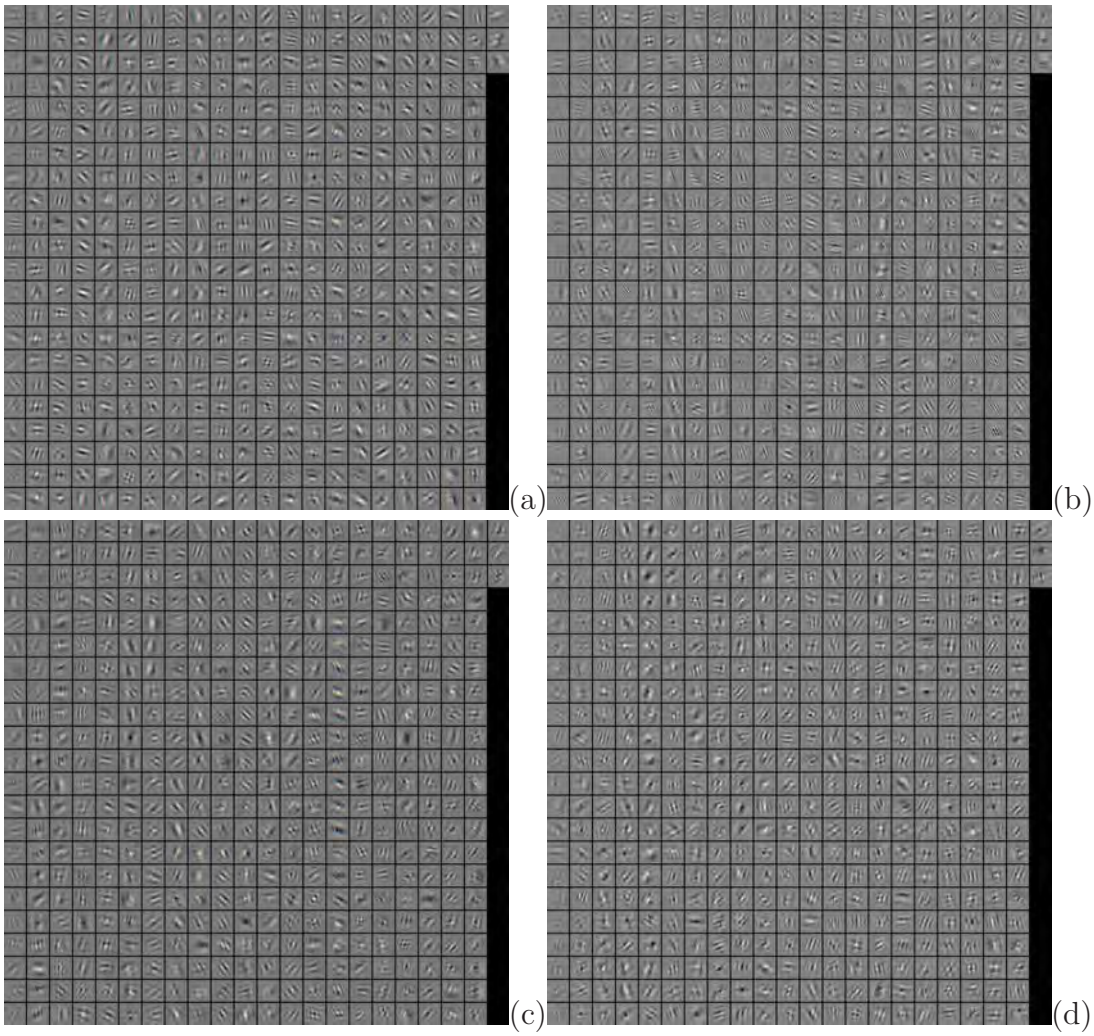
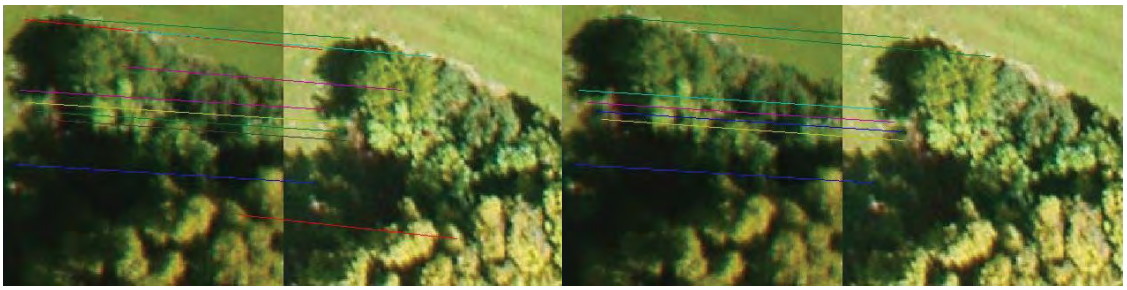


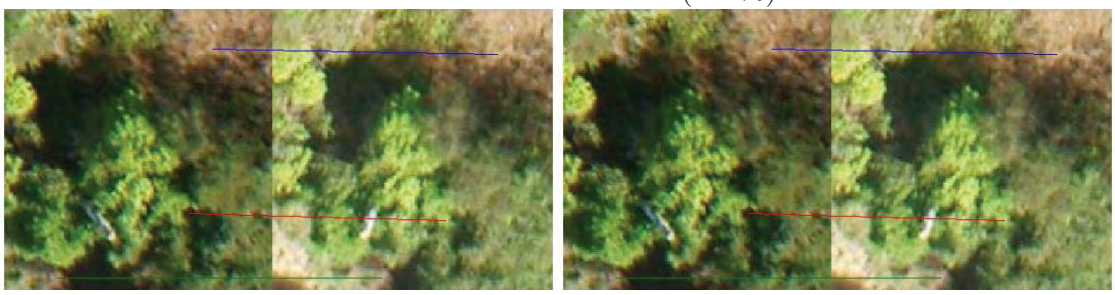
Figure A.46: Layer-2 filters on forest images (set 5) projected back into pixel space



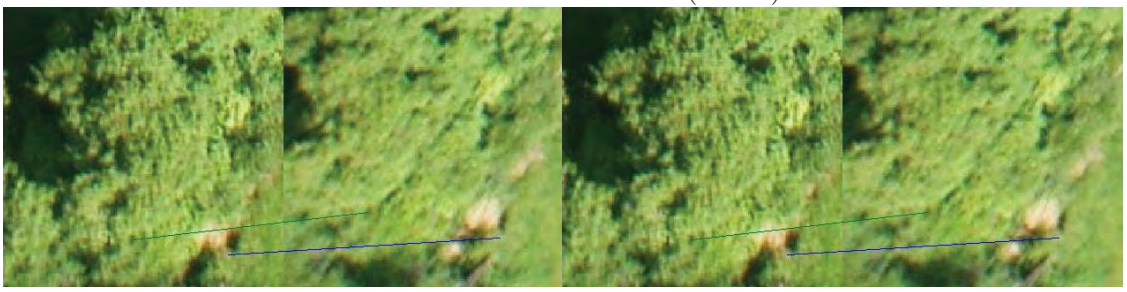
12 tentative matches | 9 (75%) inlier matches



2 tentative matches | 2 (100%) inlier matches

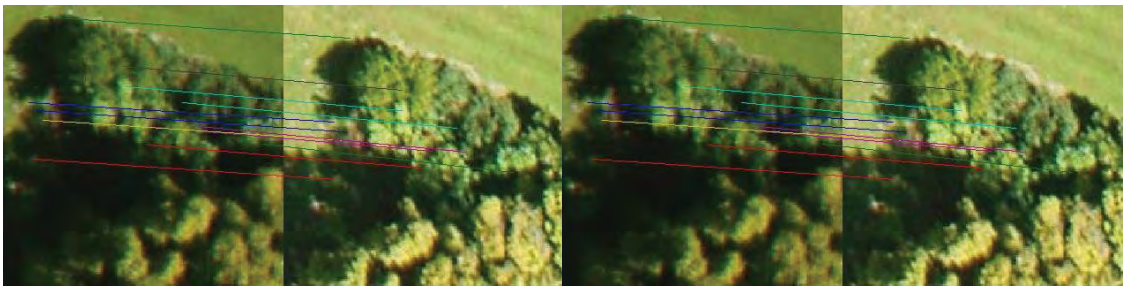


3 tentative matches | 3 (100%) inlier matches



2 tentative matches | 2 (100%) inlier matches

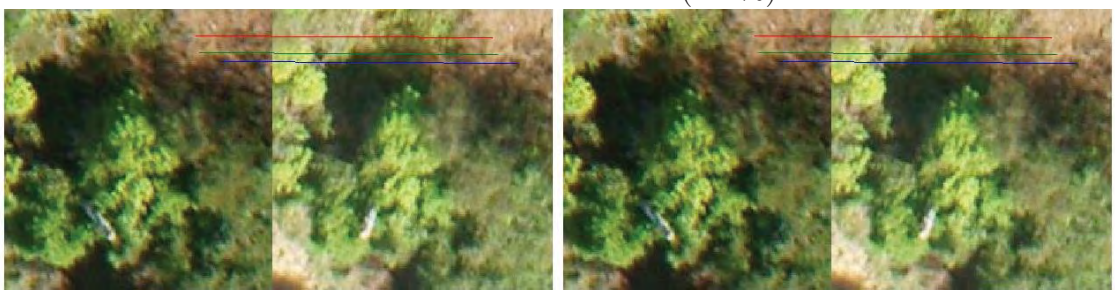
Figure A.47: Matching results of my proposed method on forest images (set 5)



12 tentative matches | 12 (100%) inlier matches



2 tentative matches | 2 (100%) inlier matches



3 tentative matches | 3 (100%) inlier matches



2 tentative matches | 2 (100%) inlier matches

Figure A.48: Matching results of SIFT detector and SIFT descriptor on forest images (set 5)