DATA SYNTHESIS FOR OBJECT RECOGNITION

BY

XI ZHANG

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science
in the Graduate College of the
Illinois Institute of Technology

Approved _____
Advisor

Chicago, Illinois
December 2017

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Large and balanced datasets are normally crucial for many machine learning models, especially when the problem is defined in a high dimensional space due to high complexity. In real-world applications, it is usually very hard and/or expensive to obtain adequate amounts of labeled data, even with the help of crowd-sourcing. To address these problems, a possible approach is to create synthetic data and use it for training. This approach has been applied in many application areas of computer vision including document recognition, object retrieval, and object classification. While a boosted performance has been demonstrated using synthetic data, the boosted performance is limited by two main factors in existing approaches. First, most existing approaches for creating and using synthetic data are application-specific and thus lack the ability to benefit other application areas. Further, such application specific approaches are often heuristic in nature. Second, existing approaches do not recognize an inherent difference between synthetic data and actual data which is termed as a synthetic gap in my proposal. The synthetic gap in existing approaches is due to the fact that not all possible patterns and structures of actual data are present in the synthetic data. To address the problems of using synthetic data and using it to better improve the performance of learning algorithm, this proposal considers general ways of creating and using synthetic data. The problem caused by the synthetic gap is studied and approaches to overcome the gap are proposed. Initial results demonstrate that the proposed approach is effective and can boost the performance to many computer vision applications including building roof classification, character classification, and point cloud object classification.

CHAPTER 1

INTRODUCTION

1.1 Overview

Computer vision is a field that focus on methods for acquiring, processing, analysing high-dimensional data from the real world, thus recognize and understand the incoming information. The recognition is based on transforming these input data to symbolic information, e.g., via decisions[5, 6, 7, 8]. A primary mission of computer vision researchers is to simulate or improve the abilities of the human vision system by processing and understanding incoming visual data. The understanding of data in computer vision research can be viewed as processing of detangling of symbolic and context information from incoming visual signals using methods and models constructed with the aid of computational methods from other disciplines, such as: geometry, physics, statistics, and artificial intelligence[9].

Recognition based on computer vision is an interesting and challenging topic that has attracted many research efforts all around the world. From simplest geometry pattern recognition to complex scene understanding based on statistical theory and artificial intelligence, the techniques used in computer vision recognition are becoming more and more advanced. In the same time, the number of applications involving computer based recognition system has expanded quickly in recent years. Nowadays, searching applications of computer vision on the internet, will return hundreds of categories that are built upon computer vision, such as: face recognition, autonomous cars, gesture recognition and so on.

Development in recent years such as deep neural networks(DNN)[10], deep belief networks(DBN)[11], convolutional nerual networks(CNN)[12] and convolutional deep belief networks(CDBN)[13], contribute to many successes in visual recognition tasks.

To mine and understand the intrinsic relation and correlation of the data, researchers

have made many efforts for designing high dimensional data features that can better characterize the nature of the data. Generally, due to the complexity of the models and the dimensions of the features, a huge number of training data is required[14, 15]. A learning process which is short of enough training data could result in many problems. One of most well-known one is over fitting where the resulting model perfectly fitted to a few training data and thus the model has large bias as with respect to the underlying ideal model. It has been proven that with a fixed number of design pattern samples, the accuracies of the model reduces as the dimensionality increases[16], the problem of which is known as the curse of dimensionality.

Data is extremely valuable and expensive. On one hand, the process of the data acquisition is very expensive and may require human labor and resources. For example, to collect real-world street view data for the computer vision benchmarks KITTI, an experienced car drive has to drive a standard station wagon through every streets in the city which equipped with two high-resolution color, a gray scale video cameras, a GPS localization system and a Velodyne laser scanner[17]. The expense of the data acquisition could go even higher when more costly data has to be collected. For example, in remote sensing, to have a detailed 3D mapping of a terrain of a region, an aircraft has to be sent with an experienced pilot and a complete set of expensive and professional remote sensing equipment. On the other hand, data labelling is much more expensive than data acquisition, and sometimes it is even impossible to get data labelled. For example, when a medical diagnosis has to be made by a model learned using hundreds of thousands of medical images, to guarantee the accuracy of prediction is in a confidence interval that is under control, the labelling of data has to be operated under advices of the physicians. Another example in which labelling is almost impossible is the recognition of the point cloud objects from street view point cloud data that is collected from real-world. A snapshot of such data is shown in Figure 1.1. We employ a learning based framework in this application, objects are categorized to 6 classes: car, pedestrian, street-light, traffic-light, tree and trash can. To have a well labelled training

set, we spend numerous hours on manipulating the data in 3D and selecting and assigning labels.

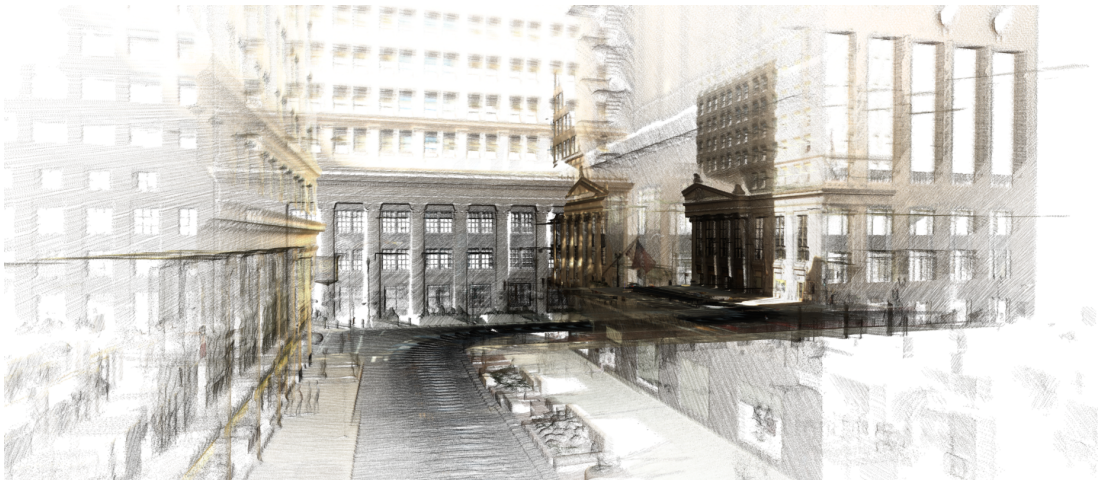


Figure 1.1. A screen-shot of street view point cloud data used in my research. There are 9195492 points that are contained in this scene.

1.2 The Proposed Solutions

Data labelling is a tedious and time consuming step of a machine learning process, and may be very expensive especially if experts are involved. To tackle this problem and reduce the cost spent on data labelling, several strategies and techniques had been proposed in last two decades.

A technique that creates synthesized data by over-sampling existing data has been extensively used in computer vision and machine learning. The most popular one among these techniques is SMOTE [18] which over-samples a minority class by using interpolation in feature space. Such interpolation gives a larger and more confident decision region to minority classes. Hence, the learner learns a less-biased boundary between the classes.

Instead of creating new synthetic data, a technique called semi-supervised learning will start from a small amount of labelled data and gradually grow and propagate to get more data labelled. In order to use unlabelled data, several assumption has to be made to the underlying distribution of data, such as: a smoothness assumption, a cluster assumption and

a manifold assumption[19]. Under this configuration, semi-supervised learning attempts to use this combination on labelled and unlabelled data were made. The idea is to improve the classification performance by propagating the label of labelled data to unlabelled data so as to expand the set of the data with labels.

All of the learning schemas and techniques mentioned above have been successfully applied by researchers in the of computer vision and machine learning for decades. However, almost all of these methods are based on an assumption that the distribution of the data is known. Such assumption usually fails unless either a data set with distribution that covers every possibility in feature space is given, which push us back to the problem of data acquisition, or the data of training set and testing set are exactly drawn from the same distribution. An example that can explain this phenomenon is given by the problem of recognizing the human body motion. To have a high accuracy recognition of human body motion, a human body model with high order of degrees of freedom is assumed with each degree of freedom representing a dimension in movement space. If the body movements in training set and testing set are totally different, (say training set containing arm movement only but testing set containing head tuning only), then the learned model has little or even no chance to recognize the testing set.

In many recognition problems, humans have prior knowledges regarding the structure and the parameter domain of the problem. Take the problem of roof style recognition from satellite images for example, the goal in this problem is to recognize the roof style of a given roof satellite image. The target labels are: hip, halfhip, gable, pyramid, gambrel, and flat. The recognition of the roof style usually is conducted by recognizing features such as ridge lines and valley lines on the roof. However, due to the degradation of image quality and occlusion caused by shadows, it is hard to detect all of these features, the cases of which are shown in Figure 1.2. To solve this problem, since we know how roof is built and we know where those ridge lines and valley lines are located on the roof, hundreds of

thousands of synthetic roofs containing these features could be created and then used to train a recognizer.

Examples of using synthetic data to assist recognition process are very common. The success of these methods and applications lie in the fact that on the one hand, the essential features of an object could be captured by a synthetic model using low level object features, such as the object silhouette, object color and object structure. Thus training a recognizer using synthetic data could capture low level features of objects and can assist the recognition process. A well known application of using a synthetic model for recognition is an object retrieval system in which users are required to provide a rough drawn sketch of underlying target[20], where the object sketches drawn by the user provide both the silhouette and shape information of an object.

A key problem of using synthetic data for recognition is the mapping between real data and synthetic ones. Reasons for such data mapping come from intrinsic differences between real data and synthetic data. Consider the roof image classification for example, the synthetic roof images are modelled as a parametric model of the combination of the roof's ridges, valleys and corners. By tuning parameters of these elements, we can generate a huge number of synthetic roofs, However by creating synthetic data, it is impossible to simulate numerous possibilities of noisy patterns in real roof images. Therefore mapping is needed to eliminate the gap between real data and synthetic ones. The mapping can be done in many ways and in different stages of the data processing. For the roof case, given a real roof, we need to match it with a synthetic roof so that the different between two is minimal. In addition we need to employ some transformation in feature space to "add" noisy patterns to synthetic data or 'remove" noisy patterns from real data. In this research, I deal with the mapping from both data space and feature space. In data space, I proposed to build a synthetic model that will minimize the distance between real data and synthetic ones. In feature space, since the distribution of the synthetic data is different from the distribution

of real data, I consider method that can make a shift for the features of either real data or synthetic data to reduce disparities between the two data. To successfully achieve these objectives I use different computer vision and artificial intelligence techniques, such as: deep learning and transfer learning.

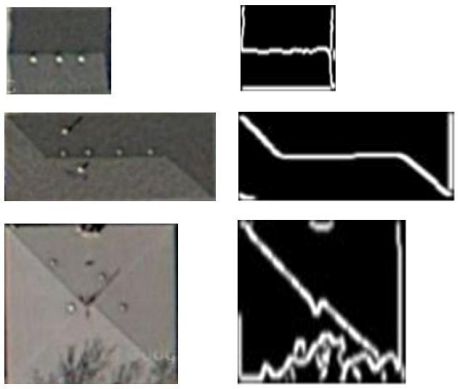


Figure 1.2. A demonstration of using synthetic template to simulate edge image of roofs used in my research.

1.3 Expected Novel Contribution

In this researches, I address recognition problems of 2D & 3D visual data using learning based methods. Generally there are two factors determining the result of a learning process.

The first factor is the learning algorithm. A successful learning algorithm should be effective and robust to unseen data. The second factor determining the learning result is the data. Data preparation usually is a pre-processing step in machine learning procedures and is often not given enough attention relatively. The quantity and the quality of the data are key factors affecting the success of a learning procedure.

It is generally accepted that the performance of a learning process could be increased if more data is available. However, data is always not enough. In my research, to solve the lack of data problem, I proposed to use synthetic data to help increase the performance of a learning process. There are various problems when it comes to data synthesis in

a machine learning procedure. To sum up, there are two major problems we need to solve. The first problem we has to solve is where to synthesize more data. The word ”where” in this context has two meanings. We first should determine in what space to conduct such data synthesis, in data space or feature space. Then we need to determine the location in each space where to add the synthetic data. These are very important questions we should really take care of, because answers to any questions will determine the complexity of the data synthesis process in the first place and the performance of the entire learning process at the end. The second problems deals with how to use the synthetic data in a learning procedure. To simulate actual data in a learning process, the synthesized data has to be as similar to actual data as possible. The synthetic data is useless unless it can truly mimic actual data. To solve these problems, my research has the following primary contributions.

1.3.1 Creation of Synthetic Data in Data Space. Machine learning algorithms usually suffers from lacking of enough training data. In existing methods researchers usually use a technique called data augmentation to increase the amount of training data. In data augmentation, new examples are generated as a perturbed version of the original data. Take image augmentation for example, to augment new images, the original images are stretched, added noises or partially cropped from the original images. Images generated in this way form a distribution in parameter and feature space which is centered at the original images. Thus, these new images do not introduce much new informations in this case. Therefore, the learner may fails to learn a more general decision boundary from the resulting data and the increase in performance will be limited.

In my research, to better simulate actual data, a series of methods that synthesize more data are introduced. Different from existing methods, my methods of data synthesis are based on usage of a centralized representation of actual data. We call such centralized representation a prototype in this proposal, because the prototype can characterize the most essential structures and patterns of actual data. The prototype in my work is modelled as a

parametric model of control points and could be learned using semi-automatic or automatic approaches. Using technique such as interpolation between existing data and randomly drawing synthetic data from a distribution computed from control points of existing data, more data could be synthesized easily.

The proposed approach has been used in many object recognition problems in my research. The preliminary results can be found in [21][22][1].

1.3.2 Learning From Synthetic Data. Directly using and learning from synthetic data may cause failure or a classification results with poor performance, even though there is enough synthetic data. This is because, the synthetic data is generated using a data prototype(parametric model) which is a simplified version of data representation. Since there is no way for a prototype to generate information such as noise and some other complicated patterns of actual data, this results in nonequivalence in terms of information contained by features computed from actual data and synthetic data. Thus synthetic data holds a different decision region with respect to actual data and the result may be poor classification performance if learned directly from these data.

A general goal of feature extraction is to minimize the difference between actual data and synthetic data by using extracted features. To achieve this goal, I propose two strategies in feature extraction that prove to be useful. First, extracted features have to be able to capture the most essential characteristics between actual data and synthetic data, thus ignoring noise all the noises and inherent patterns. Second, extracted features should be able to compensate for such additional information for synthetic data. Under these two directions, several techniques and algorithms are developed in this work.

The preliminary results of the proposed feature extraction strategies could be found in [21][22][1].

1.3.3 Elimination of Synthetic Gap. Learning a classifier from synthetic data is ex-

tremely hard, due to the following reasons. First, using a prototype to synthesize more data causes the distribution of the generated data to shift away from that of actual data. We term such distribution shift as *synthetic gap* in this proposal. The *synthetic gap* is a major obstacle in learning from synthetic data, since synthetic data may fail to simulate the potential useful patterns of real data for training classifiers. Second, since practically a small amount of labelled data may be available, it is necessary to jointly learn from synthetic and real data. The learning process must be automatically leveraged between the synthetic data and the real data.

Different from existing work, here we solve the problem from a more general perspective by eliminating the *synthetic gap* as a domain adaptation problem. A general framework based on a neural network with novel architecture is proposed. The neural network is trained as a regressor which could map synthetic data to real data and find out the potential transformation between two types of data. Thus it is able to close the gap between two data. With this mechanism synthetic data could better simulate actual data and used in a machine learning procedure.

1.3.4 Creation of Synthetic Data in Feature Space. Data synthesis through prototype in data space is very easy to implement and can be used to increase the learning performance. However, due to the nature of the methodology, it is limited to some specific problems in which it is possible to find a data prototype. For most problems in computer vision, potential data representation usually results in a very high dimensional space. Finding a prototype in such high dimensional space is a very challenging task and usually impossible.

A general solution is to directly create synthetic data in feature space. Inserting new data in feature space is hard. Because data representation in feature space is abstract that the actual meaning of the data in data space is hard to interpret. To know where to synthesize new data in feature space, requires a deep understanding of the distribution of the entire dataset. To achieve this goal, in my work, I propose a general algorithm of data synthesis

in feature space. In my approach, I compute an expected distribution of a dataset based on knowledges learned from a data down-sampling process. Then new data is synthesized to achieve or maintain an expected distribution of a dataset.

The proposed method is still under development, some more details about the preliminary results could be found in Chapter 5.

CHAPTER 2

CREATION OF SYNTHETIC DATA IN DATA SPACE

2.1 Introduction

It is generally accepted that an accurate classifier can be learned if a large and balanced labeled training dataset is available. In real-world scenarios, however, one always struggles to find adequate amounts of labeled data. Even with the help of crowdsourcing, e.g., Amazon Mechanical Turk (AMT), it is often difficult to collect a large quantity of labeled instances with high quality that is necessary for training a classifier for a real-world problem. In terms of quantity, it has been shown that amount of available training data, per class, follows Zipf distribution [23]. In terms of quality, some domains, such as analysis of satellite images(e.g. the comet images from Rosetta), require extensive and detailed expert user annotation [24]. Large volume of LiDAR point cloud data have to be labeled before they can be used to train some classifiers [1]. Such labeling process usually is very time consuming and requires expert-level labeling efforts or expensive equipments. Practically only a very limited portion of the data points can be obtained. The fundamental issue of learning from not enough data is the ability of these data to significantly compromise the performance of most standard machine learning algorithms. Most standard learning algorithm, generative and discriminative, assume that enough knowledges and features could be extracted from training data that have a balanced distribution. Therefore, when presented with a complex dataset but with less amount of data, these algorithms fail to properly represent the distributive characteristics of the data and resultantly provide unfavorable accuracies.

Technically, there is no official definition or a quantitative measure about rare data learning. It is important to understand rarity of different forms. According to [25], rarity could comes from either relative rarity which is usually related to imbalanced learning or absolute rarity which is a direct result of the nature of the data set. my method and solution

partially overlap with imbalanced learning in cases when data is imbalanced distributed, and the minority classes are viewed as rare data. More than rare data learning, methods discussed in this chapter solve more general problems in which labelling actual data maybe is very time consuming or a job must need a lot of labor works to be done.

I propose to create synthetic data and use them to solve mentioned problems in a machine learning process. The idea of using synthetic data in machine learning algorithm has a long history and is associated with the development of cognitive psychology, artificial intelligence and computer vision. The reason simply lies in the facts that first creating synthetic data is much easier when actual data could be represented as a parametric model. Second, if I view a class of objects in data space as a whole, synthetic data and actual data are both subsets of the entire dataset which may or may not overlap. Indeed synthetic data is no more than an ideal representation of actual data. As an example, I show edge images extracted from my satellite roof top images and its corresponding synthetic images in Figure 2.1. It could be observe from this figure that actual images are highly similar to synthetic ones and could be identical to synthetic ones if noises can be removed.

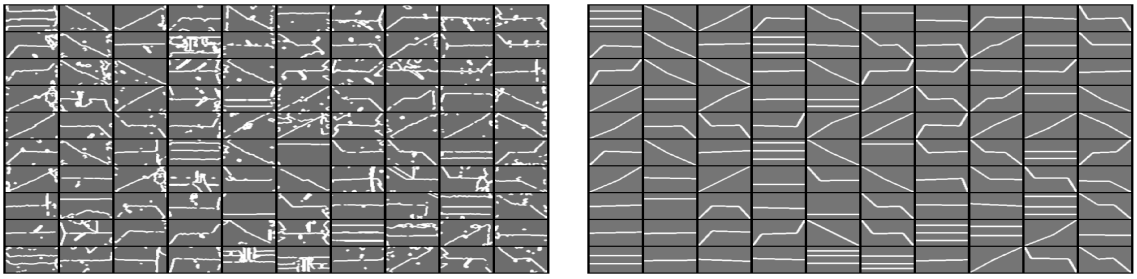
2.2 Related Work

According to my literature search, there are just a few works in the past talking about data synthesis in data space. Among these methods, most of them synthesize data in data space using tools including geometrical transformation and degradation models: In [26][27], to help off-line recognition of handwritten text, a perturbation model combined with morphological operation is applied to real data. They showed that when a moderate transformation is added to the real data, the resulting synthetic training set boost the performance. Synthetic data generation techniques has been moderately implemented in area of optical character recognition. in [28], researches and experiments were conducted to test the safeness of using synthetic data in ma- chine print text recognition problem. Following the conventional way of generating synthetic data, they model each synthetic data

as a geometrically transformed version with different magnitude of the original data. They demonstrated that classifier trained on interpolated data often but not always improved classification when tested on previously unseen samples. However, the use of interpolated data in the training sets has never worsened the results. To enhance the quality of degraded document, in [29] degradation models such as brightness degradation, blurring degradation, noise degradation and texture-blending degradation were used to create a training dataset for a handwritten text recognition problem. The synthetic minority oversampling technique (SMOTE) [30] and its variants [31][32] are also powerful methods that have shown many success in various applications. However, these previous methods are relatively limited to one particular type of dataset, whilst I propose a more general methodology of generating synthetic data by creating a parametric prototype of the data. By tuning parameters of the prototype, my method can derive synthetic data that cover almost all possibilities. A very interesting work whose intention is a bit similar to mine is proposed in [33]. In this paper, to analyse variability in point-sampled geometry, authors first assume every data comes in with incomplete information about ground true object. They then capture this uncertainty by introducing a statistical representation that quantifies for each point in space the likelihood that a surface fitting the data passes through that point. This likelihood map is constructed by aggregating local linear extrapolators computed from weighted least squares fits. The quality of fit of these extrapolators is combined into a corresponding confidence map that measures the quality of local tangent estimates.

2.3 A Brief Introduction of Proposed Methods

In my approaches, I create synthetic data in data space. Creating synthetic data in data space has several advantages. First, it is the most intuitive way to create new data, because it is easy to judge the quality of the new data. Second, creating data in data space let us understand my data more clearly and deeply, because in order to create new data, I have to understand the characteristics and structure of original data. Having these understanding,



(a) Real roof edges (b) Synthetic roof edges

Figure 2.1. Examples of (a) real roof edge vs. corresponding (b) synthetic roof edge images. The synthetic data is generated by the algorithms in Sec. 4. The examples are randomly drawn from SRC dataset.

it is more likely I will invent some better features that would boost the performance of machine learning process later on. The feasibility of my method is based on an observation that data of most problems can be converted to a parametric model, in my cases 2D and 3D models controlled by multiple control points. By adjust values of parameters I are able to derive many synthetic data.

In my research, the approach of creating synthetic data in data space primarily solves two types of problems. First, by creating more synthetic data, I solved problems of rarity learning. Because as I mentioned that both actual data and synthetic data are subset of entire dataset. By creating massive number of synthetic data, the approach enriches and compensates the part in data space where actual data does not reach, so that a more complete distribution of the data and a more tight boundary between classes of data are provided to machine learning algorithm to learn. Second, for some problems, using synthetic data will greatly reduce labor works spending on labelling actual data. Because instead of label huge number of actual data, using my method, by indicating a label of data, I can generate as many as possible synthetic data I need. For these two problems, I will use the rest of this chapter to discuss the details of creating synthetic data in my previous works. Also, I am going to talk about the problems and challenges in my future research.at the end of this chapter.

2.4 Creating Synthetic Data to Avoid Rarity Learning

Learning from rare data is very challenging, because rare data are typically much harder to identity than common data and most standard machine learning algorithms have a great deal of difficulty to detect regularities within the rare data.

Generally, there are two types of rarities in context, Much of the research on rarity relates to *rare classes*, or more generally, class imbalance. This type of rarity requires labeled examples and is associated with classification problems. A second type of rarity concerns *rare cases*. Informally, rare cases correspond to a meaningful but relatively small subset of the data, or equivalently, define a small region of the instance space. Rare cases depend only on the distribution of data and therefore are defined for both labeled and unlabeled data, for supervised and unsupervised data mining tasks. Rare cases are naturally defined by the domain and will share common characteristics. In the case of labeled data, a rare case corresponds to subconcept, or subclass, that occurs infrequently.

There are a number of problems that arise when mining rare classes and rare cases. The first problem I could encounter is lack of data. In this case, the number of examples associated with the rare class is small in an absolute sense which make it difficult to detect regularities within the rare classes/cases. Another problem could that could happen is generating a learning model by using an inappropriate inductive bias. Generalizing from specific examples, or induction, requires an extra-evidentiary bias. Without such a bias ”inductive leaps” are not possible and learning cannot occur. The bias of a data mining systems is therefore critical to its performance. One more difficulty comes with noise in training data, especially when rare classes/cases are associated with noise. Since noise data will affect the way any data mining system behaves, but what is more interesting in my research is that noise has greater impact on rare cases that on common cases. Because rare classes have fewer examples to begin with, it will take fewer ”noisy” examples to impact learned subconcept. In this case, the learner cannot distinguish between exceptional (rare)

cases and noise.

Among all of these challenges and difficulties, in my research, I primarily deal with supervised classification problem which may have *rare classes* or *rare cases* or both. By synthesize more data to create a much more balanced training set using my techniques, I demonstrate that a better performance could be obtained in many classification problems.

2.4.1 Satellite Image Roof Style Classification. Recognizing building roof styles is an important step in many applications including building reconstruction and urban planning. Such problems generally need very high quality training data. However, there is no previous dataset for such problem. Usually collecting data through online crowdsourcing, e.g., Amazon Mechanical Turk (AMT) is very expensive. So to facilitate the study, in my works [22][21], a few volunteers are invited to join the project and manually crop images from civilian level online digital maps. Manually cropping images is an tedious and time consuming task. By the end of data collection, I are able to only crop a few thousands of roof images.

For a large building which is usually comprise of multiple parts, I even develop an algorithm that reasonably segment large roof image to simple parts. I design an algorithm to decompose complex footprint. The basic idea of this algorithm is to segment the concave footprint into several convex polygons, while the cutting line should make the new polygons have less sum of variances of all inner angles in them. The reason I want to use convex polygon rather than concave polygon to recognize is generally the roof styles can be built on convex polygon much easier.

To achieve this, I assume the vertex whose inner angle is larger than as point of interested (POI). Then I need to traverse all possible angles from POI and find which cutting line leads the minimum variance of all inner angles in this segmentation. With calculated cutting lines and given footprint vertices, I can segment the polygon into basic parts. The

Algorithm 1 Footprint Segmentation

Input:
The set of footprint vertices, V;
Output:
The set of selected segmented region vertices sets, S;

Initial S;
PUSH(V, S);
if $|V| > 3$ **then**
  for $i = 1$ **to** $|V|$ **do**
    if V_i is POI **then**
      Calculate cutting line $\{V_i, D\}$, where D is the other endpoint;
      PUSH ($\{V_i, D\}, C$);
    end if
  end for
  for $i = 1$ **to** $|C|$ **do**
    for $j = 1$ **to** $|S|$ **do**
      POP (R, S), where R is the top element in S;
      Cut R by cutting line C_i into R_1 and R_2;
      PUSH (R_1, S);
      PUSH (R_2, S);
    end for
  end for
end if
return S;

algorithm shows in Algorithm 1, Let V be the set of footprint vertices, which elements has been sorted in clockwise or counterclockwise, S be the set of selected segmented region vertices sets while S is a queue. C is the set of cutting lines. R is a set of polygon vertices and R_1, R_2 are the segmented polygon vertices.

The cutting line calculation follows:

$$\sigma(i) = \sum_{j=1}^{2} \sum_{k=1}^{|A_jP|} [A_{jP(i)}(k) - \frac{(|A_jP| - 1) \times \pi}{|A_jP|}] \tag{2.1}$$

Where i is a given cutting line, $P(i)$ is the set polygons which generated by cutting line i. A_jP is the set of inner angles of j^th polygon in set P. Hence the appropriate cutting line should be $\arg\min(\sigma)$, while $\sigma(i)$ is the set of variance of cutting line i. The time

complexities of cutting line calculation and polygon segmentation are constants. Hence the total theoretical maximum time complexity is $(|V| + 2^{|C|})$, while $2^{|C|}$ always has same magnitude with $|V|$.

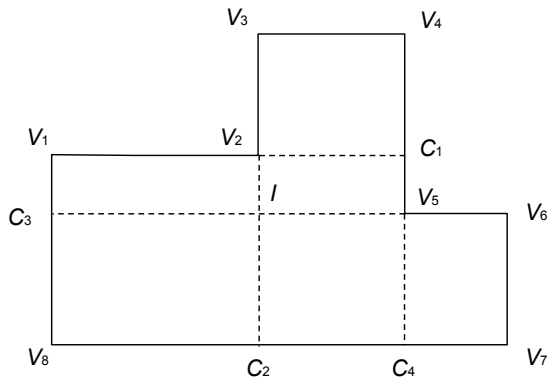
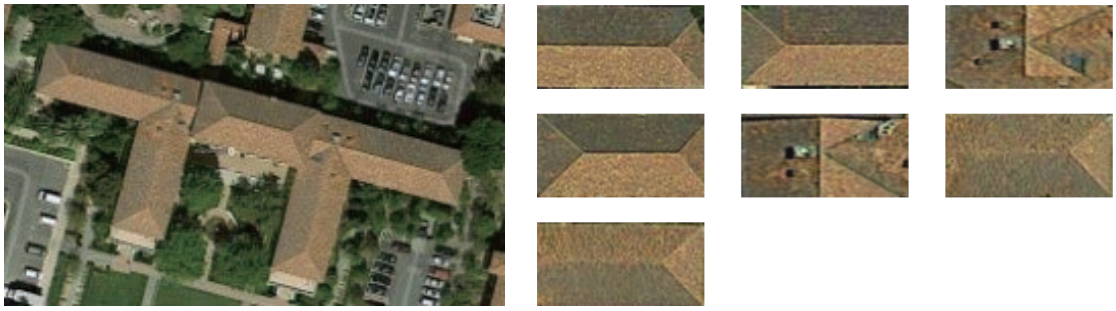


Figure 2.2. An example of input polygon vertices, cutting lines and six segmented components result.

A detailed example is shown in Figure 2.2, with input footprint vertices V. The first loop in algorithm can figure out $\{V_2, C_1\}$, $\{V_2, C_2\}$, $\{V_5, C_3\}$ and $\{V_5, C_4\}$ are cutting lines. The first cut to V by $\{V_2, C_1\}$ will get $\{V_2, V_3, V_4, V_5\}$ and $\{V_1, C_1, V_5, V_6, V_7, V_8\}$ these two polygons. Iterate all cutting lines until I get six segmented polygons.

An example of roof segmentation using the proposed approach is given in Figure 2.3 in which a building roof is divided to 7 pieces.

Later all roof top images are aligned using an approach proposed in [34]. Then all images are resized to the same dimension. This dataset is of great challenges for the task of visual analysis. First, qualities of the some satellite images are degraded because of significant image blurring occurred when capturing the satellite images. Second, roofs in these images are covered by various kinds of equipments such as air conditioners chimneys and water tanks, and most of roofs in my dataset are partially occluded by shadows cast by



(a) (b)

Figure 2.3. An example of segmentation on a complex building roof. All pieces are normalize to a same dimension.

trees and some other stuffs. Such covering and shadows are great obstacles to robust visual analysis algorithms. Examples of my data in this dataset is given in Figure 2.4

Figure 2.4. Examples of roof top images used in my work.

Furthermore, the class instances of this dataset are intrinsically extremely unbalanced that some particular types of roofs (such as gambrel and pyramid) are far less than the other types. Such unbalanced distributions of data are compared in Table 2.1.

Table 2.1. The distribution of the roof styles used in the experiments.

Styles	Training #	Testing #	Total #
Flat	1232	1748	3080
Gable	1111	1665	2776
Gambrel	156	232	388
Halfhip	268	400	668
Hip	960	1440	2400
Pyramid	133	199	332

Classification of the roof styles in the experiments are based on recognizing edges

detected from the roof images. I employed the adaptive Otsu edge detection method [35] to extract edges from the roof images. Examples of generated edge images are shown in 1.2 (a). The synthetic prototypes are then created to characterize primary structures of a roof represented by extracted edges. The way I create synthetic models is to simplify and convert roof edge images to a parametric model in which edges are connected by a few control points. By adjusting the position of the control points and drawing lines between the points, I are able to simulate different variations of roof edge images. All noises and line patterns introduced by actual roof edge images are ignored in synthetic images, which makes it easier to create synthetic images. However, without present these noises and patterns in synthetic images potentially increase the distance between actual data and synthetic data, thus lower the accuracy of classifier trained using synthetic data. I will talk about techniques and algorithms dealing with these issues in Chapter 3 and Chapter 4

For all kinds of roof styles except flat I classified in my work, I intuitively design the parametric models of these roof edge images as the ones shown in Figure 2.5.

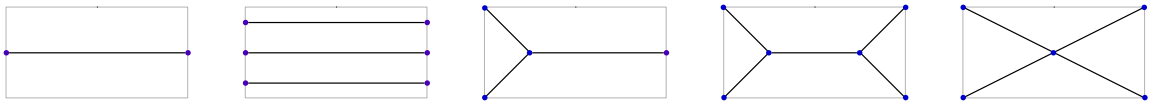


Figure 2.5. For five roof styles used in my work, control points of each style are given in above figures which are shown as blue dots.

With these basic synthetic template, generate more synthetic data is as easy as change positions of the control points. However, configuring these control points with arbitrary ranges will easily lead to a roof image that is impossible exist in real world thus introduce noises and outliers to dataset. To solve these problems, two approaches are proposed in my work [21] and [1] separately.

In [1], I assume the position of each control points in image follows a normal distribution that the set of control points: $\mathbf{P} = \{p_i\}_{i=1}^{k}$, $p_i \sim N(\mu_i, \sigma_i)$. Thus for all control

points **P** of a roof style, they follow a multi-variant normal distribution that $\mathbf{P} \sim N(\mu, \Sigma)$ where Σ is covariance matrix of **P**. To estimate parameters mean and covariance of existing data set, volunteers are invited to help us mark the control points of actual data. Then μ and Σ are computed from the marked point positions. To illustrate distribution of each control points in my roof edge templates, I rendered points' distribution in Figure 2.6. At this point, control points of a synthetic data could be randomly drawn from multivariate normal distribution $N(\mu, \Sigma)$ I just computed.

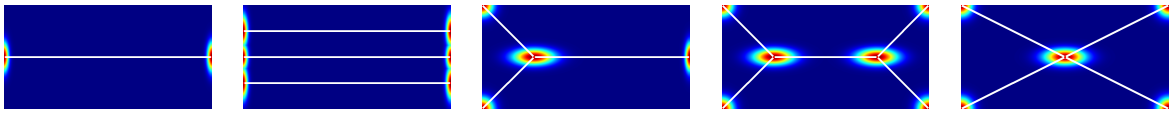


Figure 2.6. For five roof styles used in my work, control points of each style are given in above figures which are shown as blue dots.

There are several disadvantages in above approach by inviting volunteer to do the labelling and assuming underlining distribution of control points. First, manually marking control points is an annoying task that it is time consuming and easy to make mistakes. Second, assuming underlying distribution of the control points is multivariate normal distribution may be problematic, especially when the number of samples is small, such as gambrel and pyramid roof styles in my work.

So, in [21], I propose a different method to obtain positions of the control points and construct synthetic data. In [21], the synthetic data are represented as a parametric model of a set of control points and edges associated to these points in the images. From the control points, the synthetic images could be generated to simulate the real images in terms of having the same structure or a similar appearance. Initially, the control points are selected from a synthetic prototype that generalize all images in the same class. Then the locations of the control points are iteratively optimized until convergence in order to minimize the distance between synthetic images generated by control points and the real image. I annotate the control points and edges associated to them as $\mathbf{S} = \{\mathbf{P}, \mathbf{E}\}$, where

$\mathbf{P} = \{p_i\}_{i=1}^{n}$ is the set of the control points, and $\mathbf{E} = \{(p_i, p_j)\}, 1 \leq i, j \leq n$ is the set of edges connecting control points. A generalized algorithm of getting the best matching synthetic image is provided in Algorithm 2.

Algorithm 2 Get Matching Synthetic Image.

Input:

- A real image U.
- A set of control points $\mathbf{S} = \{\mathbf{P}, \mathbf{E}\}$ with all control points $p_i \in \mathbf{P}$ set to their initial positions.
- A prototype image V generated using the initial **S**.

while S is not converged **do**
  S = OptimizeControlPoints($U, V,$ **S**).
  Generate V using **S**.
end while
Generate synthetic image I using **S**.
return I.

In Algorithm 2, the OptimizeControlPoints(U, V, **S**) function is a process that searches for optimal control point locations which results in a synthetic image minimizing the discrepancy between the real image and the synthetic image. A coordinate descent framework is employed to accelerate the search process. I summarize this method in Algorithm 3.

Algorithm 3 OptimizeControlPoints($U, V,$ **S**) Case 1

Input:

- A real image U.
- A prototype of the synthetic image $\mathbf{S} = \{\mathbf{P}, \mathbf{E}\}$.
- A synthetic image V generated using **S**.

for $p_i \in \mathbf{P}, 1 \leq i \leq n$ **do**
  Update **S** by moving p_i by one unit.
  Generate V using **S**.
  if S does not reduce Dist(U, V) **then**
    Cancel the last move of p_i.
    Generate V using **S**.
  end if
end for
return **S**.

As I mentioned previously, make an assumption that the underlying distribution of control points is multivariate normal distribution may not be appropriate in some cases, especially for a case where only a few samples are given in a high dimensional space. So, I propose another technique which does not make any assumption about the distribution of

the control points, thus no parameter need to be estimated if using this method.

Algorithm 4 RNNI(I, k, N)

Input:

- Set of input images that are to be interpolated $I = \{\mathbf{S}_i\}_{i=1}^{n}$.
- The number of nearest neighbors k.
- The number synthetic data to be generated N.

Output:

- Set of generated synthetic data I.

for $1 \leq i \leq N$ **do**
  Randomly choose a integer number idx that $1 \leq idx \leq n$.
  Randomly choose a neighbor from k nearest neighbor of $\mathbf{S}_{idx}$ and call it $\mathbf{S}_j$.
  Randomly choose a floating number r that $0 \leq r \leq 1$.
  $\mathbf{P}_{new} = \mathbf{P}_{idx} + r \times (\mathbf{P}_j - \mathbf{P}_{idx})$.
  Connect $\mathbf{P}_{new}$ in a proper way to draw roof edges $\mathbf{E}_{new}$.
  $\mathbf{S}_{new} = \{\mathbf{P}_{new}, \mathbf{E}_{new}\}$.
  $\mathbf{I} = \mathbf{I} \cup \mathbf{S}_{new}$.
end for
return $\mathbf{I}$.

The synthetic data generated in this approach is based on interpolation between a sample data and one of its nearest neighbors. The interpolation is done on each existing sample. I call this method as Random Nearest Neighbors Interpolation (RNNI) to emphasize the randomness in the selection of nearest neighbor. Similar methods are used in a lot of previous works such as [3], where to facilitate character recognize, interpolation is conducted on parameters of transformation such as skew, scale and so on. my method is different from previous works that instead of compute underlying transformation, for the sake of using control points the interpolation in my approach is directly applied on data itself which makes the entire procedure easier and effective. The algorithm that generate a set of new synthetic data is given in Algorithm 4.

2.4.2 Hand Written Digital Character Recognition. Synthetic data is helpful in a lot of applications, another example that benefit from my method is hand written digital character recognition. I also validate my framework on handwritten digits dataset from UCI machine learning repository [36] which totally has 5620 instances. The handwritten digits from 0 to 9 in this dataset are collected from 43 people: 30 contributed to the training set and the

other 13 to the test set.

Generation of synthetic data in hand written digital character dataset is still done by interpolation using Algorithm 4. However, different from roof edge images, hand written digit is much more complicated, manually marking control points is almost impossible. Due to complexity of the digit character, dozens of control points must be used to guarantee the quality of synthetic digital character. Under such high dimension space of control points, it makes it hopeless and inefficient to optimize control points using approach such as Algorithm 3. Therefore, I adopt a different strategy and using an extra step to help constructing the correspondence between control points of two digital images.

To enable a interpolation with a better quality, a procedure called control point migration is adopted in my work to pair up control points of different images. The idea of the approach is to first compute an root image for all images with the same digit character which summarizes the digit character in all images. Then on this root image I could set up some control points which later could be migrated to all other images. Since for all images control points are from the same root image, their are already paired up.

A method called congealing proposed in [37] is adopted here to generate root images. In congealing, the project transformations are applied to images to minimize a joint entropy. Thus the root image actually can be considered as an average image of all images after congealing. I show all the root images in Figure 2.7.

Then control points are evenly sampled from the boundary detected from the prototype image. The control points needs to be mapped to each digit image in order to build correspondence with other images. To find this mapping I implement an approach that migrates the control points from the prototype images to destination image. Basically, given two sets of control points on two images, finding a matching between each pair of points is a bipartite matching problem, which may need some sort of features to be computed for

Figure 2.7. Root images of all kinds of digit characters.

each control points and could be solving using discrete optimization method. Additional constraint also has to be added to the solution such as matched point pairs has to keep their original order. Such constraints make the problem even harder to solve. In my method, however, the idea is very simple yet very effective that I slightly change source image a little bit towards the target image, and move the control points in the same time. my method achieve very good quality when intermediate steps are dense enough.

Figure 2.8. Illustrations of the migration of control points and intermediate synthetic images generated using control points in each step. The distance transform images of the synthetic prototype and real images are shown as the left most and right most images respectively.

This point migration algorithm is based on a series of intermediate images generated in between synthetic prototype and destination image. To generate the intermediate images, I binarize all the images and the distance transformed images[38] of the synthetic prototype and the real image are generated. Given the number of steps, an intermediate image then is generated as a binarized image of linear interpolation between two distance transformed images. In each step, the control points are snapped to the closest boundary pixels of the

intermediate image. The algorithm of OptimizeControlPoints($U, V, \mathbf{S}$) in this situation is given in Algorithm 5, I fix the number of steps to 10 in this algorithm. The procedure of control points migration is shown in Figure 2.8 and Figure 2.9. I recently notice that method proposed in [39] is very similar to my migration approach.

Algorithm 5 OptimizeControlPoints($U, V, \mathbf{S}$) Case 2

Input:

- A real image U.
- A prototype of the synthetic image $\mathbf{S} = \{\mathbf{P}, \mathbf{E}\}$.
- A synthetic image V.

1: $steps = 10$.
2: Compute distance transform image of U, V as U', V'.
3: **for** $i = 1$ to $steps$ **do**
4: $\quad I = (1 - \frac{i}{steps})U' + \frac{i}{steps}V'$.
5: $\quad I$=Binarize(I).
6: $\quad$ Update **S** by snapping to the closest boundary pixel on I.
7: **end for**
8: Set the status of **S** to be converged.
9: **return S**.

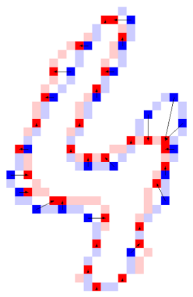


Figure 2.9. Illustrations of the migration of control points for character 4. The control points are migrated from root image (blue) to destination image (red) and arrows are used to indicate points moving direction.

Once all images get their control points migrated from root image. More synthetic images could be generated using Algorithm 4.

2.5 Creating Synthetic Data to Avoid Manual Labelling One of requirement for a successful machine learning algorithm is abundant labelled data with high quality. Such dataset usually requires a long time and a lot of labor work to be done, which make the

labelling work very expensive. In some cases, it is even impossible to finish labelling such as the problem of point cloud roof style recognition I solved in [1].

in [1], I proposed a learning based roof style classification algorithm using aerial LiDAR point clouds. The proposed approach is able to classify complex roof styles which may be composed of simple roof styles including: curve, flat, gable, hex, hip, mansard, pyramid, shed, gambrel, dome and unknown. All roof styles classified in my work are listed in

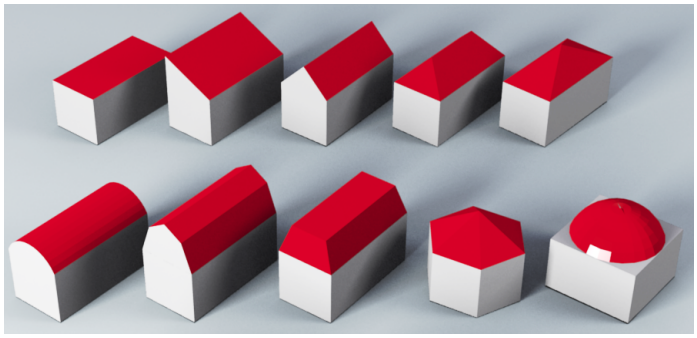


Figure 2.10. Roof styles I classified in [1]. From top to down, left to right: flat, shed, gable, hip, pyramid, curve, gambrel, mansard, hex and dome.

This method can easily accommodates more new roof styles by re-training of the new dataset with the new roof styles added. Because my method is implemented based on a bag of works schema.

The difficulty in this work is to produce a codebook of important roof points. This codebook can then be used for a bag of words recognition. Learning these codebook using unsupervised learning is most straight forward. However, unsupervised learning is often misguided easily by the data and detects uninteresting patterns within the data.

Instead, I propose to integrate existing knowledge of roof structure and cluster the points of target roof styles into several semantic classes which can then be used as code words in the bag of words model. I use synthetic variants of these code words to train a semantics point classifier. Thus, I manually represent the codes in the codebook as semantic

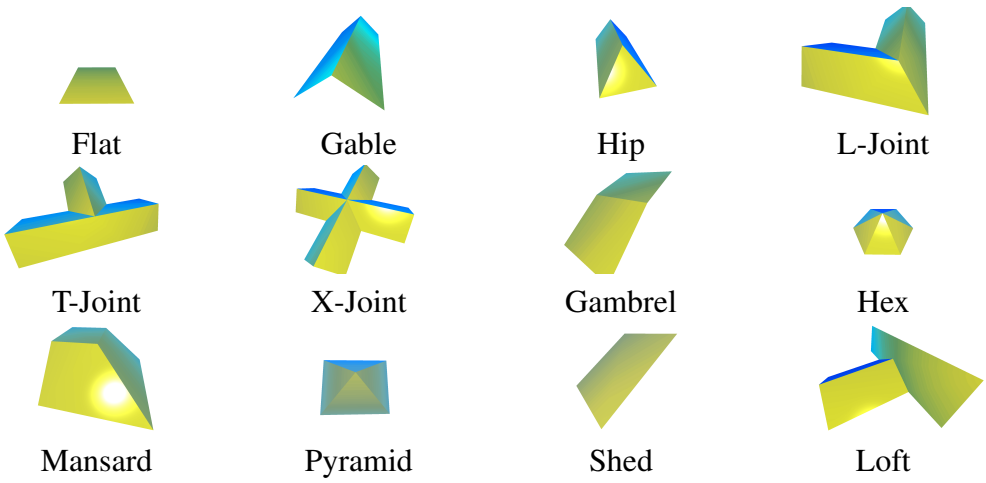


Figure 2.11. Illustration of all roof base models that are used in this work.

parts of the roof structure produced by manually analysing roof style models. I then learn a point semantics classifier using a large amount of synthetic variants of roof styles. There are 33 codes in the codebook used in this work. The key points that are used to produce the code words are shown in Figure 2.13.

As can be observed in Figure 2.13, some roof styles shown in Figure 2.11 are used to produce multiple key points. The synthetic variants of the codes are derived from these based models in two ways. First, I generate roof models by changing the parameters of the roofs including height, width and slope. Since it is inaccurate and unreasonable to assume any distribution for these parameters and I want to generate models that can cover as much as possible cases in real world, I evenly draw these parameters from their given ranges. I present the size derivation of each base roof in below:

Flat: A little slop is allowed in my flat roof, so I change the angle between flat surface and ground plane from 0 degree to 10 degree, as an exception, distribution of roofs in between these two angles are even.

Gable: The opening width of gable roof is set to be 10 meters, I change two parameters for gable. Firstly, the median ridge is shifted, starting from center location and end at 2.5 meters from center. Secondly, by fixing opening width, the dihedral angle between slop

surface and ground is changed from 20 degree to 75 degree, where mean is 30 degree.

Hip: In addition to how I derive from gable roof, the angle between side slop surface and ground plane is changed from 20 degree to 75 degree where mean is 30 degree.

L-Joint: For each of two branches, derivation of gable is applied.

T-Joint: For each of two branches(vertical and horizontal), derivation of gable is applied.

X-Joint: For each of two branches which cross with each other, derivation of gable is applied.

Gambrel: For two slop surface, the angle between surface tangent direction and z direction is changed. Suppose the angle of upper surface with z is a_0 and angle of lower surface with z is a_1. Using 45 degree as mean, a_0 and a_1 are changed from 20 to 75 degree, subject to $a_0 - a_1 \geq 20$ degree.

Hex: Synchronously change the slop angle of side face from 20 degree to 75 degree, where mean is 30 degree.

Mansard: Synchronously change the slop angle of two side face from 20 degree to 75 degree using 30 degree as mean.

Pyramid: Similar to Hex

Shed: Change slop angle from 20 degree to 75 degree using 30 degree as mean.

Loft: For slop surface, Change slop angle from 20 degree to 75 degree using 30 degree as mean and for gable part, please refer to derivation of gable.

In addition to size variation, I also consider erosions of roofs, because it is common in my data to have eroded roofs due to unknown reasons. Four versions with different magnitude of erosion are derived using approach described in [40] for each generated roof which are shown in Figure.

Figure 2.12. Illustration of all semantic points used in my work, red sphere on each roof represent the typical location of semantic point, the color bar under each image correspond to all color labels used in previous demo.

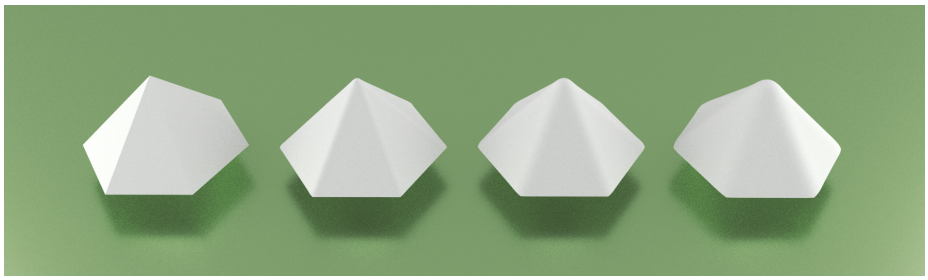


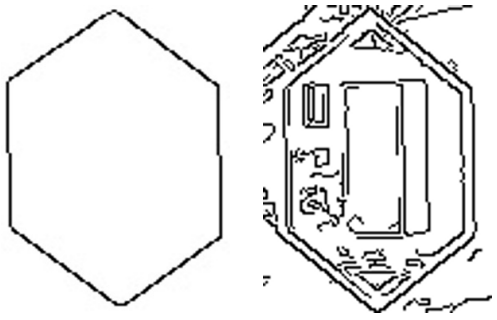
Figure 2.13. Illustration of erosion in my data. Three different levels of filtering are applied to each roof. In total, including original version, four versions are used in my work.

CHAPTER 3

LEARNING FROM SYNTHETIC DATA

3.1 Introduction

Directly use and learn synthetic data will most likely cause a failure or performance going downward in machine learning even the amount of synthetic data is far enough. Because, synthetic data is generated using parametric model which is a simplification of actual data. Thus, intrinsic patterns and noises owned by actual data do not appear in synthetic data. An example of actual roof edge image and corresponding synthetic roof edge image in my work [2] is shown in Figure 3.1. It could be seen from the figure that synthetic building roof image characterize the primary structure only which is a boundary of hexagon. However, the actual roof image in real world comes with more edges extract from inside and edges extracted from other buildings outside the boundary. Some times, such differences could be caused by method of data collection itself, such as a the dataset I used in my work [1], that it is very hard for synthetic data to compensate these difference which is not necessary either. Actually, it is almost impossible to simulate every possible noise or inherent patterns in synthetic data using parametric model, which is not the intention of creation of parametric models in my method.



Actual Synthetic

Figure 3.1. An example of actual roof edge image and corresponding synthetic roof edge image in my work [2] is shown in above figures.

Such differences between actual data and synthetic data is very hard to eliminate in

data space and will cause a shift of synthetic data domain from actual data domain. Thus any machine learning framework learned from these synthetic data will most likely failed to work for actual data. Such shift is defined as a *Synthetic Gap* in my work [21] which will be discussed in detail in Chapter 4. It is necessary for a learning procedure to remove or at least weaken such discrepancy in order to have a successful recognition result later on.

Information contained by actual data and synthetic data is not equivalent. Actual data could either contains more information in terms of noises and inherent patterns, or contain less information due to data loss such as occlusion in images. Synthetic data used in my work are created based on several parametric prototypes. The prototypes are composed of and simplified to contain the most basic and essential structure of actual data.

A general goal of feature extraction is to minimize the difference between actual data and synthetic data in extracted features. There two ways to achieve such goal First, extracted features has to be able to catch the most essential characteristics between actual data and synthetic data, thus ignore all the noises and inherent patterns contained . Second, extracted features should be able to compensate these additional information for synthetic data. Under these two directions, several techniques and algorithms are invented in my works.

3.2 Related Work

Most of existing works create synthetic data as a perturbation of original data [3][41][42][26][27][28]. Synthetic data under this domain is generated by either deforming original data using geometric transformation such as scaling, rotation, slant, shrinking ans so on or degradation model such as adding noise, erosion, removing parts and so on. Such data synthesis is very easy to implement in practise although, results in very limit data coverage in data space. Thus, generated synthetic data still carry the same amount of information and stay in the same disjunct as original data.

So far as I know, just very a few existing works treat synthetic data specially either in feature extraction or learning procedure. One of earliest works dealing with this problem is [3]. In this work, authors pioneer to use synthetic data in character recognition. They assume characters undergo a series of geometric transformation defined by them. To simulate real data, their recognizer is trained using synthetic data using their transformation. An interesting setup in their procedure is they apply a set of predefined inverse perturbations to the input image and are expected to include the true perturbation that actually made the input image different from its standard pattern. The corresponding inverse image will be very close the original standard pattern and could be easily recognized by some known method if an inverse perturbation actually corresponds to the true perturbation. Therefore in their learning pipeline, each inverse image is submitted separately to a conventional recognition system, the output score of which is then compared to others. It is clear that among the scores, the one corresponding to the true perturbation can be expected best. Since each score is attached to a class, the recognition scheme is in fact a by-product of the reversing process. A demonstration of this system is show in Figure 3.2. Similar ideas are implemented in [42] and [43] in which degradation model or transformation model are trained and tested separately.

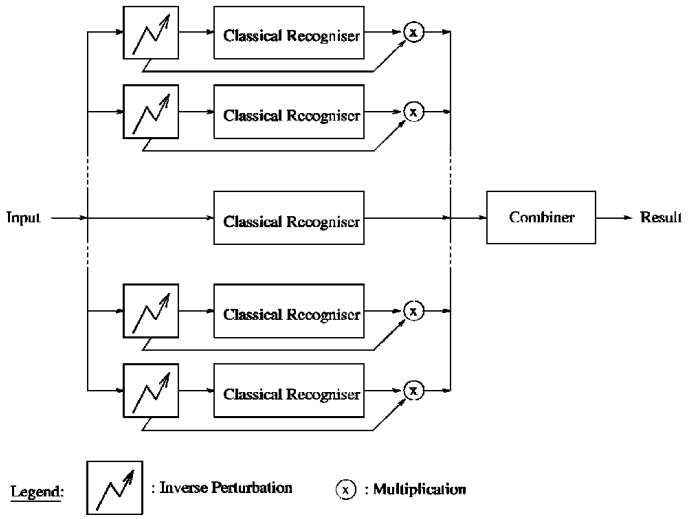


Figure 3.2. The schema of perturbation-based recognition of [3] is shown above.

Another strategy to handle the inherent difference between synthetic data and actual data is to use highly abstract features which to some extent could ignore divergence between two data and are able to extract the most common and essential characteristics from the two data. I notice such idea is applied in [41] where to build a optical character recognition (OCR) system, a Kolmogorov complexity distance is used to measure the similarity between two data. Kolmogorov complexity measures the amount of information contained by each data which is done by computing the highest compression ratio of data when being compressed. By using this feature, feature extraction process ignore the concrete differences between data and focus on most essential information contained in the data.

Most of existing works generate synthetic data in feature space. Most of techniques are invented to generate and learn synthetic data in feature space. I put the discussion of using synthetic data in feature space to Chapter 5 and use that chapter to present more details of algorithms and strategies.

3.3 Features to Ignore Additional Information in Actual Data

3.3.1 Roof Style Classification. In my work [22], my goal is to classify actual satellite roof edge images using synthetic edge images I created in Chapter 2. The synthetic data is created as connected line segments on image to represent roof main structures such as ridge and valleys. In actual roof edge images, due to low image quality in original source images and noises coming from for example shadows, occlusions and small objects on roofs, the edges representing main roof structures are composed of intermittent short segments and a lot of random size circles. Due to these defect, traditional image features such as Histogram of Gradient (HOG), Local Binary Pattern (LBP) do not work very well on minimizing the difference between actual data and synthetic data.

To facilitate a better learning from synthetic satellite roof edge images, I propose a

Table 3.1. Precision and recall of hip (HIP), gable (GBL), flat (FLT) and half hip (HHP) styles classified by Random Forest is shown in above table. I use red font to show highest value among results.

	Real Roof		Combination	
	Precision	Recall	Precision	Recall
HIP	0.953	0.823	0.963	0.814
GBL	0.877	0.901	0.886	0.882
FLT	0.784	0.977	0.745	0.959
HHP	0.893	0.439	0.967	0.509

new image feature called Histogram of Oriented Rays (HOR). HOR works by tracing the longest continuous ray in each direction centerred at each pixel. A few parameters are set to enable HOR to be tolerant to small gaps along the ray and a small range of perturbation of the pixels on the ray. Similar to HOG feature, HOR uses a histogram to save information from all orientations around a pixel, shown in Figure 3.3. The exact algorithm of HOR is given in Algorithm

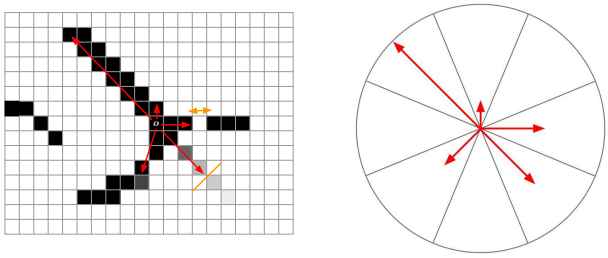


Figure 3.3. Illustration of HOR feature.

To evaluate the performance of HOR, a test set containing 1783 roof images including flat, hip, gable and half hip roof styles is tested. I use random forest as a base learner and compare results to features including HOG, Shape Context (SC), LBP and so on. A same feature merging rule is used in all tests that each feature descriptor has 5 pixels per cell, 2 cells per block and 50% overlapping between blocks. The precision and recall obtained using HOR feature are show in Table 3.5.

I show the accuracies of each feature using my testing data classified by the classifier, which trained by the combination of my training dataset and synthetic models. The

Algorithm 6 Histogram of Rays

Input:

Given point p on gray scale image I;
Number of search orientation D;
Searching threshold T;
Mimimum gap length G;
Length of HOR feature vector L;

Output:

The HOR feature vector V of point I;

for $i = 1$ **to** D **do**
  Terminator $count = 0$;
  Calculate pixel list on direction $\frac{D}{2\pi} \Rightarrow IDX$ from p;
  Index $j = 0$;
  while $count < G$ **do**
    if $IDX_{j+1}/IDX_j < T$ **then**
      $count++$;
    else
      $count = 0$;
    end if
    $j++$;
  end while
  $V_i = j$;
end for
Sampling V from length D to L;
return V;

accuracies is presented in Table 3.2. The precisions calculated by using HOR and HOG combination for hip, gable, flat and half hip(93.1%, 95.5%, 98.2% and 66.7%) shows a better result than using HOR(89.8%, 95.0%, 96.8% and 63.2%) and HOG(80.5%, 88.2%, 95.4% and 59.7%) separately. The results show that for current edge detection approach and training method, HOR alone, or HOR combined with HOG, are considered to be more appropriate as the input feature for roof style recognition system.

There is no roof labelled as half hip in the subset weevaluate of my sponsors dataset. The only one I obtain is in Figure 1, where captured in Stanford campus. I cant say the accuracy for half hip style is 100% based on this ideal same. The final accuracy of roof in complex footprint recognition is 95.2%, 84.6% and 91.7% for flat, hip and gable styles.

Table 3.2. The comparison of accuracies between HoG (HOG), Shape Context (SC), HoR (HOR) and their combinations by running Random Forest approach.

	HIP	GABLE	FLAT	HALFHIP
HOG	0.805	0.882	0.954	0.597
SC	0.350	0.828	0.959	0.140
HOR	0.898	0.950	0.968	0.632
LBP	0.000	0.986	0.631	0.000
HOR+HOG	0.931	0.959	0.982	0.667
HOR+SC	0.619	0.891	0.959	0.436
HOG+SC	0.752	0.959	0.945	0.474
SC	0.743	0.869	0.963	0.509

3.3.2 Satellite Building Image Retrieval. An extension of classic chamfer matching algorithm is proposed in my another work [2]. In this work, I deal with a image retrieval problem for edge images extracted from building roof top. Different from roof top images I used in other works, roof top images in [2] are extracted from artificial building footprint, thus, the edge images contain only boundaries of a roof. An example of target roof image and synthetic roof image is given in 3.1.

The task in this work is to retrieve the most matching actual roof edge image given a synthetic roof edge image. Challenges come from different aspects. First, actual edge image are generated by extracting edges from satellite images, thus edge images could contain a great number of edges from other unrelated objects in image. In addition, due to lighting condition, buildings in image could either be too dim or shadowed by dark areas. Second, synthetic edge image basically are polygons representing the boundary of a building. These polygons give a very approximate shape about an actual building, thus, the polygon could be very different from the actual building roof top.

In order to be resistant to all kinds of defects in actual roof edge images, I extend the classic chamfer matching feature and design a new chamfer matching that is much more robust when facing an image retrieval problem such us the one in my work.

The idea of using Chamfer matching for image registration was first introduced by

Barrow *et al.* [44] where they try to find the model of a coastline in a segmented satellite image. Since then many variations of the Chamfer matching have been introduced.

An important variation of the Chamfer matching by Borgefors [45] uses a hierarchical Chamfer matching algorithm (HCMA). This algorithm uses an image pyramid in searching for the optimal position of a template. The search for an optimal position is made in different resolution levels of the pyramid by using a representation of the distance image. The optimization of the objective function is done by discretizing the transformation parameters and stepping through them in each pyramid level. The speed of HCMA can be improved by modifying the computation of the distance transform image and by selecting the starting search position [46]. An HCMA algorithm based on interesting points has been used in [47] where a parallel computation scheme of Chamfer matching is discussed. The selection of interesting points in this work is done through a dynamic threshold scheme guided by a histogram.

A Chamfer matching algorithm which is based on multiple features was introduced by Gavrila et al.[48],[49], where it is proposed to use edge orientation as a feature. An orientation channel is created for each feature and a distance transform image is generated for edges in the channel. This method uses a hierarchical scheme for matching multiple templates with an image in which similar templates are grouped at different levels.

Shotton et al.[50] use the Chamfer distance with an additional cost which measures the mismatch of edge orientations given by the average difference in orientation between template edges and the closest edges in target image. Instead of explicitly formulating a separate term of orientation mismatch, the orientation difference is generalized in the computation of the Chamfer distance[51]. A comparison between shape context matching and Chamfer matching is conducted in [52], where results show that using the Chamfer matching is faster than matching using shape context and that global matching using Chamfer matching is better than using shape context.

Chamfer Matching

Let U and V be two binary edge images, where U is the target image and V is the template image. Let $\{u_j\}_{j=1}^{m}$ and $\{v_i\}_{i=1}^{n}$ be the edge pixels in these images respectively. Let $U(u_j)$ denote the value of image U at location u_j.

In chamfer matching, I seek a correspondence between $\{u_j\}$ and $\{v_i\}$ under transformation W. Assuming that the transformation W between the template and target images is rigid with translation T and rotation R, a template edge pixel v_i is transformed into the target image by using following expression:

$$W(v_i;\ R,T) = R \cdot v_i + T \equiv v_i^U \tag{3.1}$$

Given a distance metric $d(\cdot)$, I can solve for the transformation parameters T and R by minimizing the total distance:

$$D(U,V;\ R,T) = \frac{1}{n} \sum_{1 \le i \le n} d(v_i^U, u_j) \tag{3.2}$$

where u_j is the closest target edge pixel to v_i^U in the sense of the distance metric $d(\cdot)$.

The computation of $d(v_i^U, u_j)$ can be done in linear time using the *distance transform* of the target image[53]. Denoting the distance transform image of U by $U_{\mathcal{DT}}$, I can write Equation (3.2) as:

$$D(U,V;\ R,T) = \frac{1}{n} \sum_{1 \le i \le n} U_{\mathcal{DT}}(v_i^U). \tag{3.3}$$

Various types of distance transforms can be produced using different distance metrics $d(\cdot)$. In [53], city-block distances are used and a two-step linear algorithm is proposed to compute the distances. Euclidean distances approximations are provided by

Borgefors[38], Montanari[54], and Danielsson[55]. An efficient squared Euclidean distance computation algorithm is described by Felzenszwalb[56], Felzenszwalb's algorithm can be generalized to compute other distances.

Extended Chamfer Matching

I extend the Chamfer matching in several ways.

First, I extend the basic Chamfer matching and obtain additional robustness by jointly minimizing the spatial distance between pixels and the misalignment of edge orientations. A similar distance metric is used in [51] where a linear representation of edges is generated to model the edge orientation. The proposed approach avoids the need for a linear representation of edges by computing the edge orientation directly from the images.

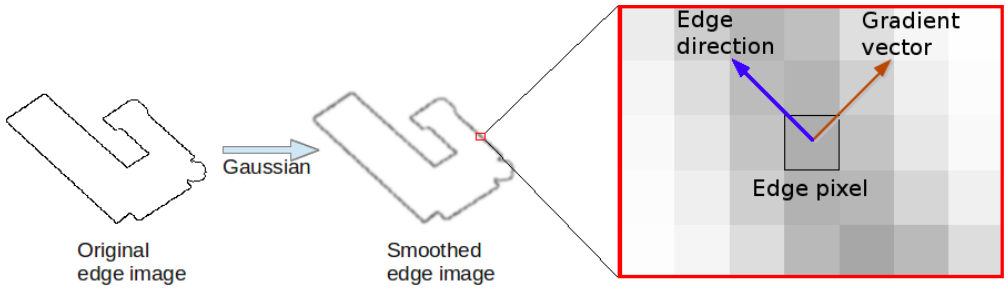


Figure 3.4. Computing edge orientation in the binary template image.

The edge orientation at each location is computed as a vector perpendicular to the gradient vector at that location. In the target image, the distance transform image $U_{\mathcal{DT}}$ is used to compute the edge orientation at each location. Specifically, given a transformed location v_i^U, its edge orientation $\vec{v}_i^U$ is computed as a vector perpendicular to the gradient vector $\nabla U_{\mathcal{DT}}(v_i^U)$. To compute the edge orientation in the binary template image V, I create a gradient vector field around edges by smoothing V using a standard Gaussian kernel. The edge orientation vectors $\vec{v_i}$ are then calculated as vectors perpendicular to the gradient vectors obtained from the gradient vector field. An example of the edge orientation computation in V is shown in Figure 3.4.

Having the edge orientation computed in both the target and template images, the distance between pixels v_i^U and u_j is computed by:

$$d(v_i^U, u_j) = \lambda U_{\mathcal{DT}}(v_i^U) + (1-\lambda)(1 - \| \cos(\alpha_{v_i^U}) \|) \tag{3.4}$$

where λ is a weight factor that controls the importance of orientation mismatch, $\cos(\alpha_{v_i^U}) = \langle \vec{v_i^U}, \vec{v_i} \rangle$ measures the orientation mismatch, and it is assumed that $\vec{v_i^U}$ and $\vec{v_i}$ are normalized.

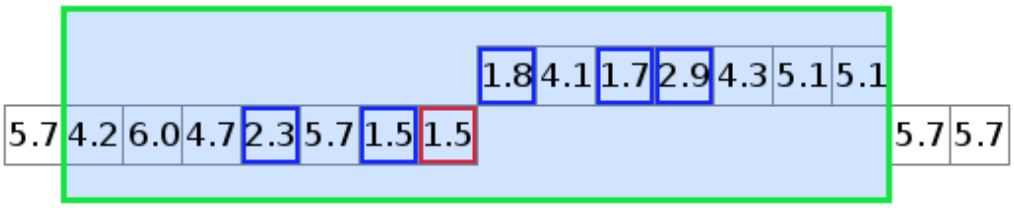
A squared Euclidean norm is used in the first term of Equation (3.4), which gives a larger penalty to mismatched pixels. A method to generate the squared Euclidean norm distance transform in linear time is described in [56].

Edge Distance Variance

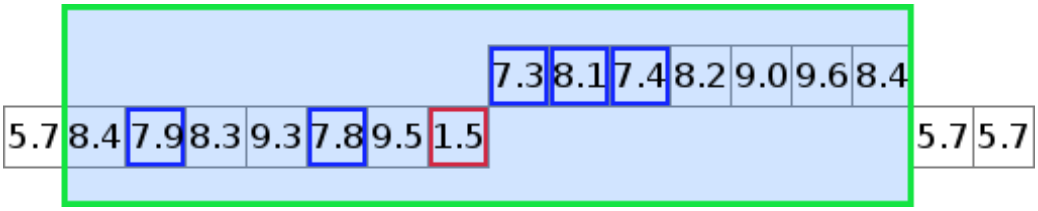
To have a robust matching result, the matching error of a pixel v_i should not solely depend on $d(v_i^U, u_j)$. This is because it is possible that v_i^U will be an incorrect edge pixel in the target image that happened to have a small error. Based on this consideration, I argue that to get a confident measurement of the matching error it is necessary to take into account the matching error of neighbouring pixels.

Given the transformed template edge pixels $\{v_i^U\}_{i=1}^n$, I find for each pixel v_i^U matching scores for the p closest transformed template edge pixels. To estimate contextual matching error of $d(v_i^U, u_j)$ at location v_i^U, the q pixels with lowest matching error, where $q < p$, are selected. I denoted these q pixels as $\{v_k^U\}_{k=1}^q$. I then calculate the variance of the distance $d(v_i^U, u_j)$ of the q selected pixels:$\varphi(v_i^U) = \frac{1}{q}\sum_{k=1}^q (d(v_k^U, u_j) - \bar{d})^2$, where $\bar{d}$ is the average matching error of the q neighbors.

The distance variance $\varphi(v_i^U)$ provides a more stable assessment of the matching



(a) Modified distance = 1.5 × 0.55 = 0.82



(b) Modified distance = 1.5 × 2.54 = 3.82

Figure 3.5. Example of the modified distance measure. In both cases I use a neighborhood of size $p = 13$, and select the lowest $q = 5$ neighbors. While the distance at the pixel is the same (1.5), the modified distance measure in the bottom example is higher due to the larger distance to neighbors.

result at v_i. The parameters p and q control the size of the contextual information used in the computation. A larger p leads to more contextual information included while q helps in excluding outliers. In my experiments, I set $p = 13$ and $q = 5$. Since p and q are constants, the asymptotic time complexity of the algorithm is not affected by them. From a practical point of view, to save time when searching for the p closest neighbors at each v_i, I generate and maintain a list of p closest neighbors for each v_i before the matching begins.

As I would like to give preference to distance measures with small distance variance, I modify the distance metric in Equation (3.4) by multiplying it by a factor which is proportional to the distance variance:

$$d_{\varphi}(v_i^U, u_j) = d(v_i^U, u_j) \times (1 + \varphi(v_i^U)) \tag{3.5}$$

Figure 3.5 shows an example of the modified distance computation.

The significance of multiplying the distance variance in Equation (3.5) lies in sev-

eral aspects. First, the distance measure is no longer solely dependent on each pixel's individual matching distance as the distance now considers the relationship between a pixel and its best matched neighbours. Consequently, I expect each pixel and its connected neighbours to have a small distance. Second, the variance in Equation (3.5) makes it much easier to separate well matched pixels from mismatched ones by increasing the width of boundary that separates the well matched pixels and mismatched ones.

I tested the proposed approach on two datasets, where each contains 1000 building models selected from a San Francisco (SF) and a Chicago (CHI) urban area.

The four parameters in Algorithm **??** are set as follows: $\theta = 50\%$, $t_s = 5$, $t_\alpha = 15$, and $t_\varphi = 0.8$. By setting these parameters, I allow for at least 50% of template pixels to be inliers in the computation of $D(U, V;\ R, T)$; the accepted matching error of obtained transformation has to be within error of 5 pixels in terms of spatial distance and 15 degrees in terms of orientation difference; finally 0.8 average distance variance is used to rule out outliers during Chamfer matching.

In my experiments, I test the accuracy of my algorithm in estimating the optimal transformation during alignment. I manually labelled the ground truth locations of the building roofs in satellite images. The accuracy of the result is measured by the ratio of overlap between the bounding box of the ground truth roof mask and the one transformed by the proposed algorithm. Note that simply measuring the Root Mean Square error (RMS) between the aligned targets and templates is not accurate as it is sensitive to broken edges and incorrect alignments.

I first test the proposed Chamfer matching without using global constraint. I divide my evaluation into two parts. In the first part, the evaluation is done on all the buildings of both datasets using the same λ. I then change λ between 0 and 1 to compare the robustness of the proposed approach, and evaluate the relative importance of the spatial and angular

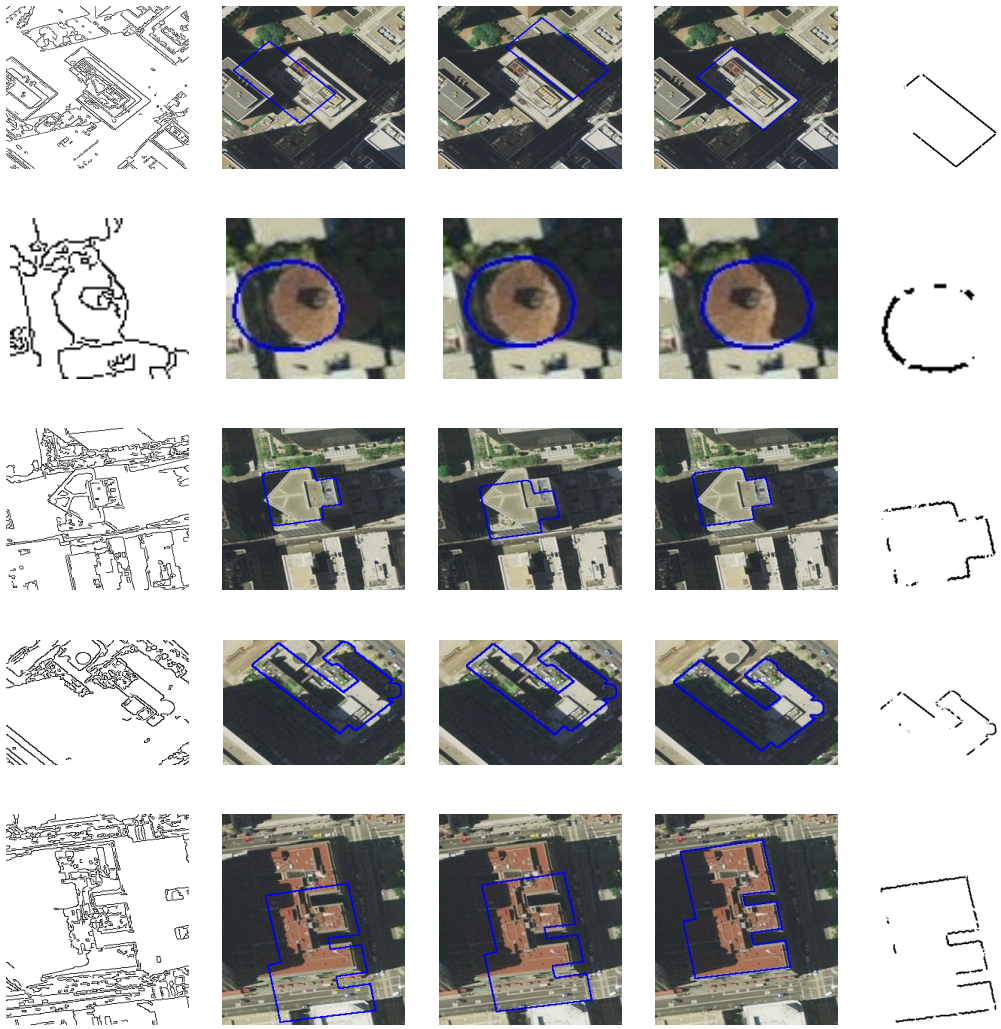
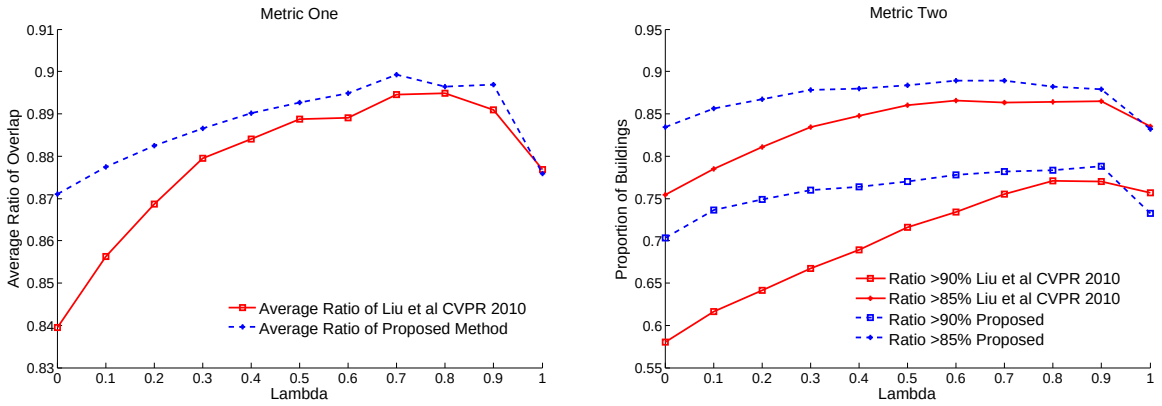
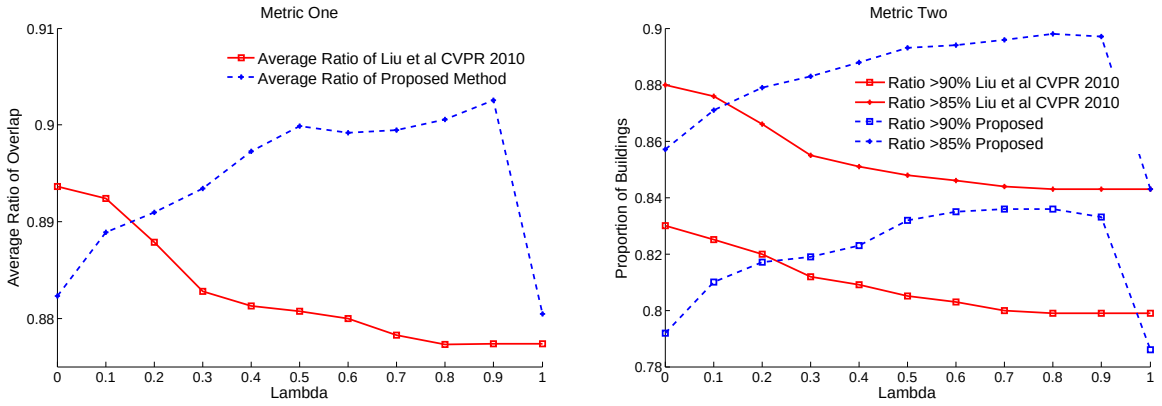


Figure 3.6. Results of matching partially occluded buildings. (a) Edge images detected from target image. (b)-(d) Results of the CM algorithm, the results of the DCM algorithm and the results of the proposed algorithm, respectively. (e) The pixels selected for computing the average error during the matching process.



San Francisco



Chicago

Figure 3.7. Experimental evaluation results on the San Francisco (upper row) and Chicago (lower row) datasets. The left and right columns show different evaluation metrics.

terms. In the second part, to assess the performance of the proposed algorithm on matching occluded objects, I specifically run tests on building images in which the target buildings are partially occluded. In all the tests, I compare my results with that of the directional Chamfer matching (DCM) proposed in [51] and the basic Chamfer matching (CM). Note that the result of CM is a special case of the DCM algorithm when $\lambda = 1.0$.

Two kinds of metrics were used in my evaluation. In the first metric, I compute the average ratio of overlap area in the two datasets. In the second metric, I consider only results with area overlap above a certain rate (85% and 90%).

In the first test where all the buildings were included, it could be observe in Figure 3.7 that my algorithm generates better results compared with DCM and CM at almost every setting of λ. In both datasets, my algorithm maintains a similar and consistent performance, while DCM performs differently on the two datasets.

Table 3.3. Accuracy results.

		Metric One	Metric Two $\geq 85\%$	Metric Two $\geq 90\%$
SF	Proposed	91%	0.94	0.90
53 Buildings	DCM	63%	0.33	0.28
	CM	35%	0.20	0.18
CHI	Proposed	80%	0.81	0.78
74 Buildings	DCM	51%	0.21	0.20
	CM	32%	0.16	0.13

To test proposed algorithm's strength in finding matching targets under partial occlusion, I specifically test and evaluate the algorithms' performances on 53 buildings from San Francisco and 74 buildings from Chicago which are significantly occluded by shadows. I use a parameter of $\lambda = 0.7$ in this test. The results are shown in Table 3.3. As can be observed, the proposed algorithm achieves 80% accuracy on both datasets compared with about 50% accuracy when using DCM and 30% accuracy when using the original Chamfer matching. Some examples of matching incomplete buildings are shown in Figure 3.6.

I finally tested the improvement obtained by the global constraint for $\lambda = 0.7$.For both data sets, I obtain at least 2% improvement in terms of accuracy. The comparisons of alignment results with and without global constraint alignment are given in Table 3.4.

Table 3.4. Results of alignment using global constraint. The number on the right side of the arrows show the results obtained by alignment using the global constraint.

	Metric One	Metric Two $\geq 85\%$	Metric Two $\geq 90\%$
SF	$89\% \rightarrow 91\%$	$88\% \rightarrow 90\%$	$78\% \rightarrow 79\%$
CHI	$89\% \rightarrow 92\%$	$89\% \rightarrow 91\%$	$83\% \rightarrow 85\%$

3.4 Features to Compensate Additional Information for Synthetic Data

There are cases when ignoring additional patterns in actual data are very difficult. An typical example is given in my work [1] where the exact structure of actual data is hard to be recognized thus there is no way under this scenario to find corresponding geometry of synthetic data. Therefore, rather than designing feature to neglect additional patterns on the side of actual data, I am going to learn additional patterns on actual data and design features which are able to encode these information in the computation.

This done from different aspects in [1]. First, features in this work are designed to characterize spatial and context information of local point.

• Spatial Features

Spatial features take into account the neighborhood of each point. So the spatial features are powerful to describe geometry characteristics in the point's neighborhood and so assist in producing a distinct characterization of it. Let p_i be a key point. Let N_i be a neighborhood of the point, containing neighbors in a radius of μ (0.6 meters in my experiments). I denote the spatial features at p_i using $\mathcal{S}_i$.

i. Eigen Features (EF). I compute the eigenvalues: $\lambda_1, \lambda_2, \lambda_3$ of the covariance matrix of neighbors N_i centred at p_i. I then add following features for p_i: λ_1, λ_2, λ_3, $\lambda_3 - \lambda_2$, $\lambda_2 - \lambda_1$,

$\lambda_1/(\lambda_1+\lambda_2+\lambda_3)$, $\lambda_2/(\lambda_1+\lambda_2+\lambda_3)$, $\lambda_3/(\lambda_1+\lambda_2+\lambda_3)$, yielding 8 features in total. These features capture the non-planarity of points around p_i.

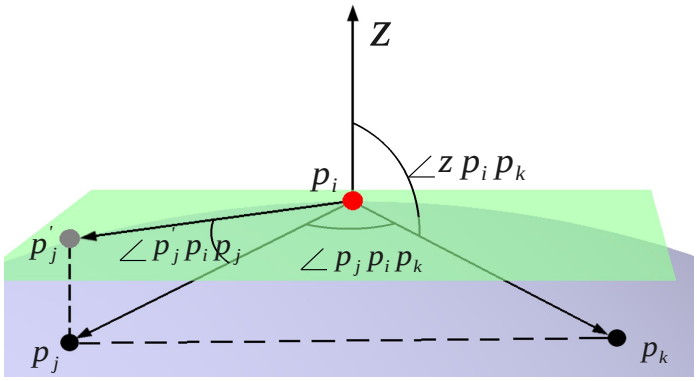


Figure 3.8. Illustration of the frame I used in the computation of the shape distribution features.

ii. Point Feature Histogram (PFH). These are point cloud features based on histogram which are described in [57]. Taking into account computational efficiency, 3 subdivisions of the features range are used in the feature histogram which yield $3^3 = 27$ features.

iii. Shape Distribution Features (SD). Shape distribution [58] measures global geometric properties of an object by representing object features as a probability distribution. Shape distribution is invariant to translation, rotation and scale and is highly informative in matching objects. Using the ideas of shape distribution I construct four features:

1) A2: Measures the angle between two vectors composed by two neighbors $p_j, p_k \in N_i$ and p_i, shown as $\angle p_j p_i p_k$ in Figure 3.8. A2 feature is computed for all pairs of points chosen from N_i.

2) Az: Measures the angle between the z direction and a vector pointing from p_i to one neighbor p_k, shown as $\angle z p_i p_k$ in Figure 3.8. All points in N_i are used to compute this feature.

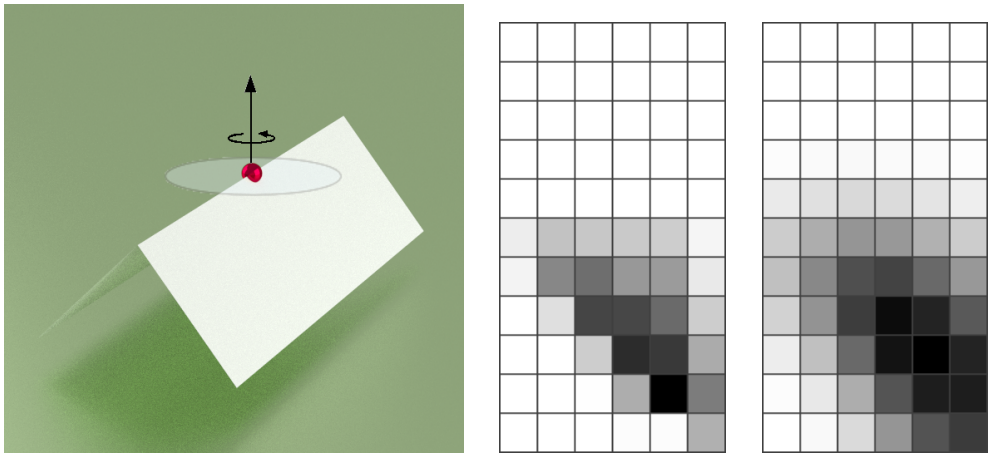
3) D2: The is the feature D2 as described in [58]. It measures the distance between any two neighbors of p_i, an example is shown as $\|p_j p_k\|$ in Figure 3.8. D2 is computed for all

pairs of points chosen from N_i.

4) Dt: Measures the angle between p_i's tangent plane and a vector pointing from p_i to another neighbor p_j, shown as $\angle p'_j p_i p_j$ in Figure 3.8, where p'_j is the projection of p_j on tangent plane. All points in N_i are used to compute this feature.

I use a histogram with 10 bins to represent each of the features described above. In total there are 40 SD features that are computed.

iv. Spin Image. Spinning around the z direction, I compute a spin image [59] with $6(width) \times 11(height)$ dimensions. Totally 66 features are contributed by spin image.



(a) (b) (c)

Figure 3.9. Example of applying Gaussian smoothing to the spin image features. (a) The red dot shows the location where spin image features are computed. (b) Generated spin image without smoothing. (c) Result of spin image after random disruption.

Synthetic data are created as locally perfectly planar or intersection of perfect flat surfaces. However, this usually is not a case in actual data where surfaces are locally irregularly bumpy and disrupted by outliers and noises. To better encode these irregularities in the spatial features, I learn these bumpiness from actual data and simulate them in synthetic data when computing above spatial feature.

In my work bumpiness is defined as the distance a point moving up and down along

vertical direction. To model the bumpiness in my work, I make following assumptions. First, bumpiness is correlated to slope of local surface where points on flat surface are less likely to have bumpy effect than points on the slope. Second, bumpiness follows a normal distribution on direction of point's normal for points on the similar slope surface, shown in Figure 3.10. These assumptions though are quite heuristics, it make sense to have them in this work. Because laser signal decay much faster when bouncing back from slope surface thus the noise and outlier variance is larger for points on a surface with larger slope.

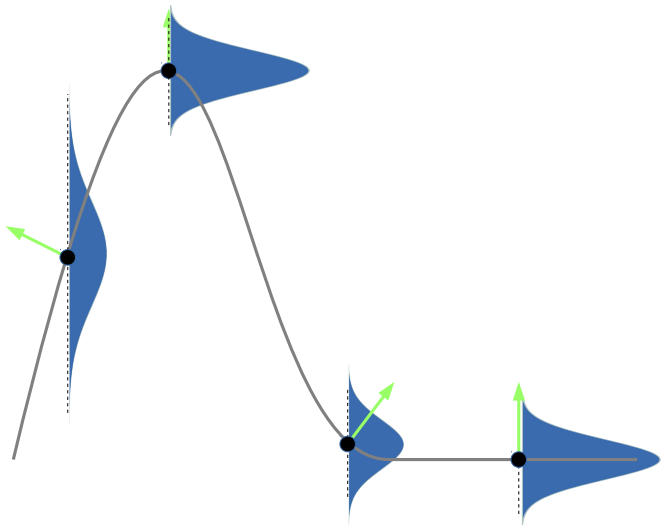


Figure 3.10. Illustration of distribution of bumpiness at points with different slope.

To model bumpiness, I assume entire data collection process is a stochastic process in which I assume for points with similar slope, point offset caused by noise is a Gaussian noise with zero mean $\Xi = \{\xi_i\}_{i=0}^{n}, \Xi \sim N(0, \sigma_i)$. For each point p_i, the noise associated with the point p_i is then computed as vertical distances to plane surfaces fitted by each of its neighbors. Denoting local plane surface fitted at point $p_j, p_j \in N_i$ as $F(p_j)$, where N_i is set of neighbors of p_i. Then the offset of p_i can be computed as a expectation of vertical distance to planes fitted by all its neighbors:

$$\text{Exp}(D(p_i)) = \sum_{p_j \in N_i} D_{F(p_j)}(p_i)\Phi(\|p_i - p_j\|) \tag{3.6}$$

where $\Phi(\cdot)$ is a weighting function that $\sum_{p_j \in N_i} \Phi(\|p_i - p_j\|) = 1$, and $D_{F(p_j)}(p_i)$ represents the vertical distance from p_i to a plane fitted locally at p_j.

To model the relationship between point offset and its slope, the slope of local surface is computed as the dihedral angle between point normal and horizon. Given the maximum and minimum of the slope of all points, a histogram $H_s = \{h_i\}_{min(\text{Exp}(D(p_i)))}^{max(\text{Exp}(D(p_i)))}$ is built, Suppose k is the number of bins in the H_s and I use $H_s(i), 1 \leq i \leq k$ to denote the centring value of point slopes in each bin. Later, H_s is voted using $\text{Exp}(D(p_i))$. Then variance $\text{var}(h_s(i))$ is computed for distances within each bin $h_s(i)$. Therefore, I could have k tuples of $(H_s i, \text{var}(h_s(i)))_{i=1}^{k}$ for which I assume a polynomial relationship exists. Thus, a polynomial modal can be fitted to the tuples. An example of polynomial model with degree 2 is shown in Figure 3.11.

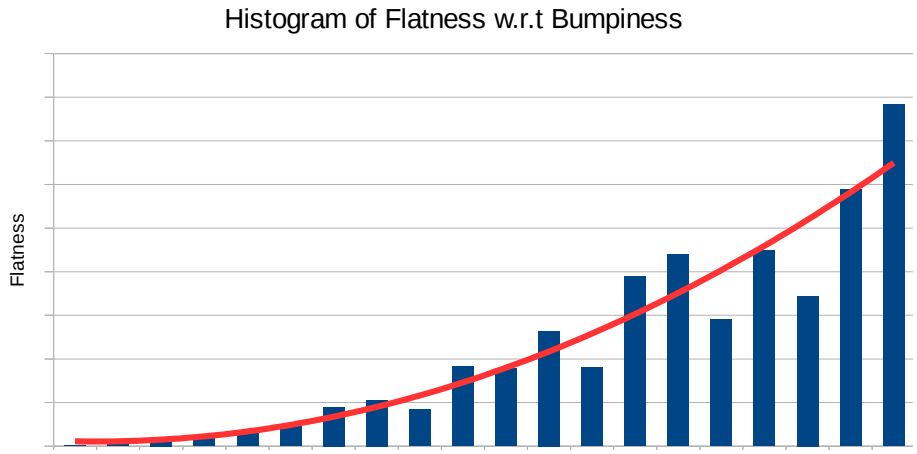


Figure 3.11. Illustration of the histogram H and the quadratic function fitted on it.

To this end, I have obtain the polynomial relationship between point slope and the variance of distribution of point offset with which I can encode this randomness in features generated from synthetic data. Given a synthetic roof, I first compute slope of every point on the roof, after which a adjustment of points location along vertical axis is applied to each point using polynomial model just obtained. Such position adjustments have been applied to all features introduced previously. An example in Figure 3.9 shows a comparison of spin

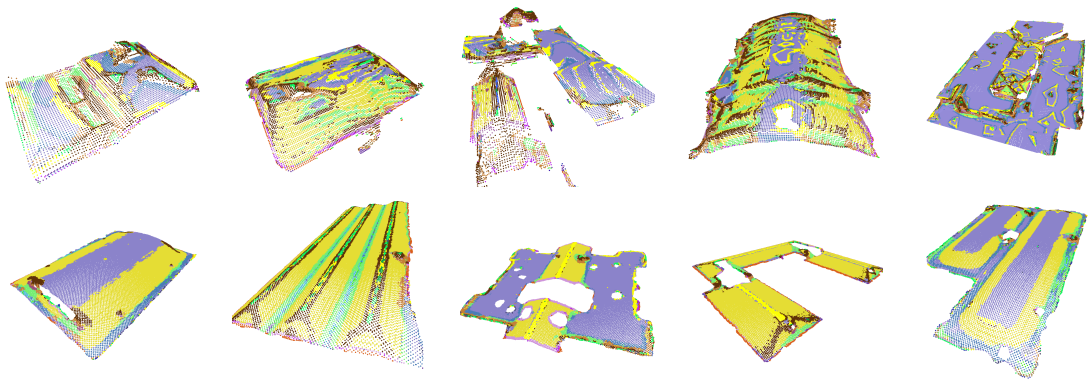


Figure 3.12. Point classification results obtained by running my point type classifier on dataset one (top row) and the dataset two (bottom row). The colors of points corresponds to the color bars of the point in Figure 2.13.

image between using point adjustment and without.

The classification of roof style in this work has two stages. In the first stage, I classify points according to their semantics. A random forest is used in the first stage. The classifier produces a probability measure for each of the 33 code words. Thus given a point p_i I have $\{P(l_j|\mathcal{S}_i)\}_{j=1}^{33}$, where $\mathcal{S}_i$ is the spatial feature set at p_i and l_j is the j-th label $j \in [1, 33]$. There are two parameters in setting the random forest: the number of trees ρ_0 in the forest and the number of features ρ_1 that the random forest can choose at each node. Assuming that the dimension of the input feature vector is ϵ, so I set $\rho_0 = 100$ and $\rho_1 = \sqrt{\epsilon}$. A histogram $\mathcal{H} = \{h_j\}_{j=1}^{33}$ is used to count the frequency of the points in a roof. Given points of a roof $\mathcal{P} = \{p_i\}_{i=1}^{n}$, each bin h_j of $\mathcal{H}$ is computed as:

$$h_j = \frac{1}{n} \sum_{i=1}^{n} P(l_j \mid \mathcal{S}_i, \mathcal{C}_i), \quad 1 \leq j \leq 33 \tag{3.7}$$

An example of the classification of roof points by the proposed approach is shown in Figure 3.12. As can be observed in this figure, context features produce more regular and confident result. After convergence, the histogram $\mathcal{H}$ is used as the bag of words features of the roof.

The roof style classifier uses the bag of words features $\mathcal{H}$. I use a random forest classifier to classify each roof style into one of 9 possible roof styles. To accommodate various kinds of point cloud degradations in different datasets, the roof style classifier is trained using a real roof dataset. I set the two parameters of the random forest as: $\rho_0 = 100$ and $\rho_1 = 6$. To avoid bias towards a particular roof style and create a balanced number of training data among different styles of the roofs, I super sample the training data using the SMOTE [60] algorithm.

The proposed approach has been tested on two datasets. In the first dataset, there are 3290 buildings that were extracted from Chicago urban area. In the second dataset there are 3290 buildings that were extracted from a San Francisco urban area. The two datasets have different characteristics and kinds of degradations.

In the first dataset, roof points are irregular and the shape of the roof is decayed to some extent. This is due to the fact that roofs in this dataset were originally produced from a highly down-sampled aerial LiDAR. The roof points in the second dataset have relatively low resolution and uneven distribution across the roof surface. Both datasets contain roof points of building only. The roofs were labelled to one of 9 target roof styles according to their appearances. A roof is labelled as UNKNOWN when either the roof is not recognizable or its roof style can not be categorized into one of the 8 styles. When a roof is composed of multiple styles, I label the roof according to the style of the largest component. The distribution of roof styles in the two datasets is shown in Figure 3.13.

To evaluate the performance of my approach and measure the improvement in using the synthetic model when generating the codebook, I compare the roof style classification results obtained by my approach to the results obtained by the K-Means (KM) and the Gaussian Mixture Model (GMM) algorithms for generating the codebook.

I evenly divide the dataset into two parts, one part is used as a training set, while

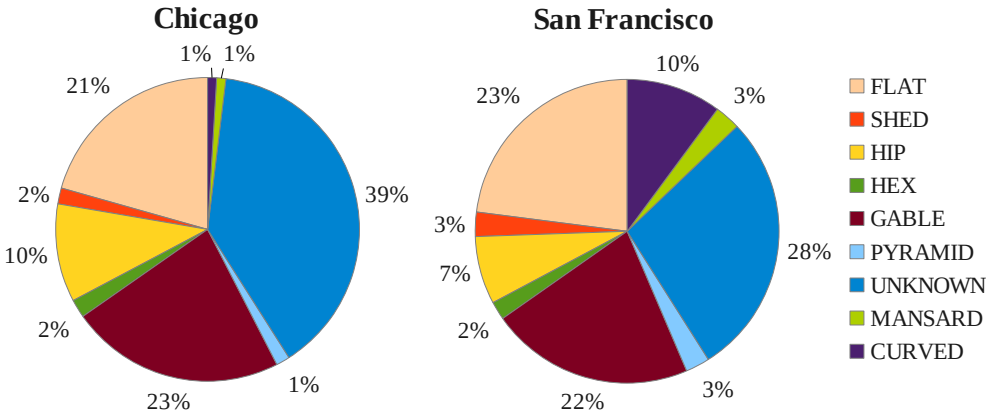


Figure 3.13. The distribution of the roof styles in the two datasets.

Table 3.5. Precision and recall of Gaussian Mixture Model(GMM), K-Means(KM) and my approach are shown in above table. I use red font to show highest value among results obtained by these three approaches.

	Chicago						San Francisco					
	Precision			Recall			Precision			Recall		
	GMM	KM	Ours	GMM	KM	Ours	GMM	KM	Ours	GMM	KM	Ours
FLAT	0.87	0.85	0.92	0.88	0.90	0.90	0.88	0.88	0.86	0.93	0.93	0.93
SHED	0.94	0.94	0.91	0.57	0.64	0.78	0.87	0.91	1.00	0.43	0.68	0.68
GABLE	0.62	0.67	0.71	0.86	0.84	0.88	0.57	0.61	0.65	0.71	0.69	0.77
HIP	0.65	0.63	0.63	0.16	0.36	0.37	0.55	0.61	0.70	0.22	0.28	0.31
HEX	0.87	0.86	0.93	0.87	0.81	0.90	0.90	0.80	0.92	0.83	0.66	1.00
PYRAMID	0.83	0.66	1.00	0.20	0.16	0.37	0.87	0.83	1.00	0.87	0.93	1.00
MANSARD	1.00	0.75	1.00	0.25	0.18	0.31	0.50	0.66	1.00	0.05	0.11	0.41
CURVED	1.00	0.93	1.00	0.87	0.93	1.00	0.71	0.70	0.74	0.77	0.71	0.79
UNKNOWN	0.84	0.85	0.97	0.88	0.85	0.90	0.62	0.59	0.66	0.63	0.64	0.66
Average	0.85	0.79	0.89	0.62	0.63	0.71	0.72	0.73	0.84	0.60	0.63	0.73

the second part is used as a testing set. In the case of the KM and GMM algorithms, the codebook is generated using the training set. For each approach, the bag of words features of all buildings are then computed using its own codebook. I set $k = 30$ in the K-Means algorithm and $k = 27$ in the Gaussian Mixture Model algorithm as these were the two configurations that provided the best accuracy for these approaches. The roof style classifier of each approach is then trained and tested using its own bag of words features. The results of these three approaches in terms of precision and recall are shown in Table 3.5. F-Score results are shown in Figure 3.14.

As can be observed, the proposed approach performs better for almost all roof

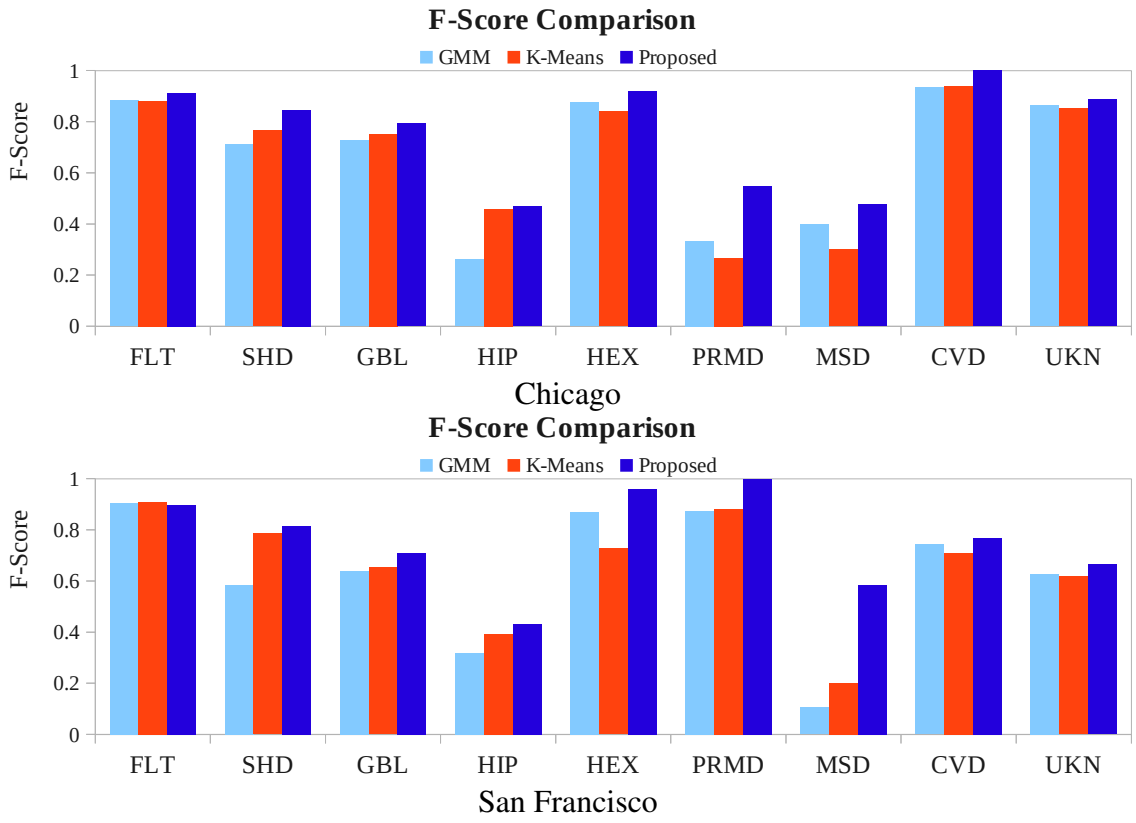


Figure 3.14. The comparison of F-Score on the two datasets by running KM, GMM and the proposed approach.

styles. Considering the roof styles of PYRAMID and MANSARD, I observe that the performance of KM and GMM is limited by the fact that the training set does not contain many examples of such roofs. In contrast, the proposed approach uses synthetic models and so is not affected by a small set of training examples.

The confusion matrices of the proposed approach on the two datasets is shown in Figure 3.15. As can be observed, the proposed approach performs well in classifying most of the roof styles. Errors mostly come from the confusion between the hip and gable roof styles, which are actually similar. Indeed, visual examination confirms that in my dataset many hip roofs resemble gable roofs due to shape erosion.

The hardest part in the classification is the recognition of the unknown roof styles, because there is no regular pattern (bag of words features) for roofs in this category. I

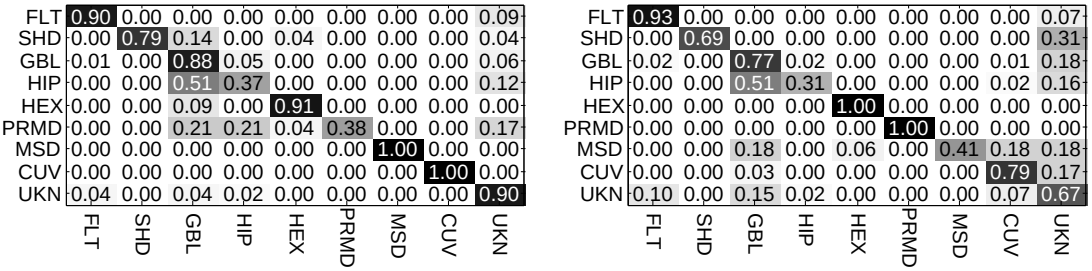


Figure 3.15. The confusion matrix of obtained by the proposed approach on two datasets.

observe that I get accuracy above 90% in the Chicago dataset and accuracy above 67% in the San Francisco dataset.

I evaluate the performance of the proposed approach as a function of the training set size. The F-Score obtained by the proposed approach as a function of training set size is shown in Figure 3.16.

As can be observed, the proposed approach achieves stable performance on most of the roof styles. For roof styles with relatively low score, the curves present are ascending which could suggest that a better score can be obtained once more training data is included.

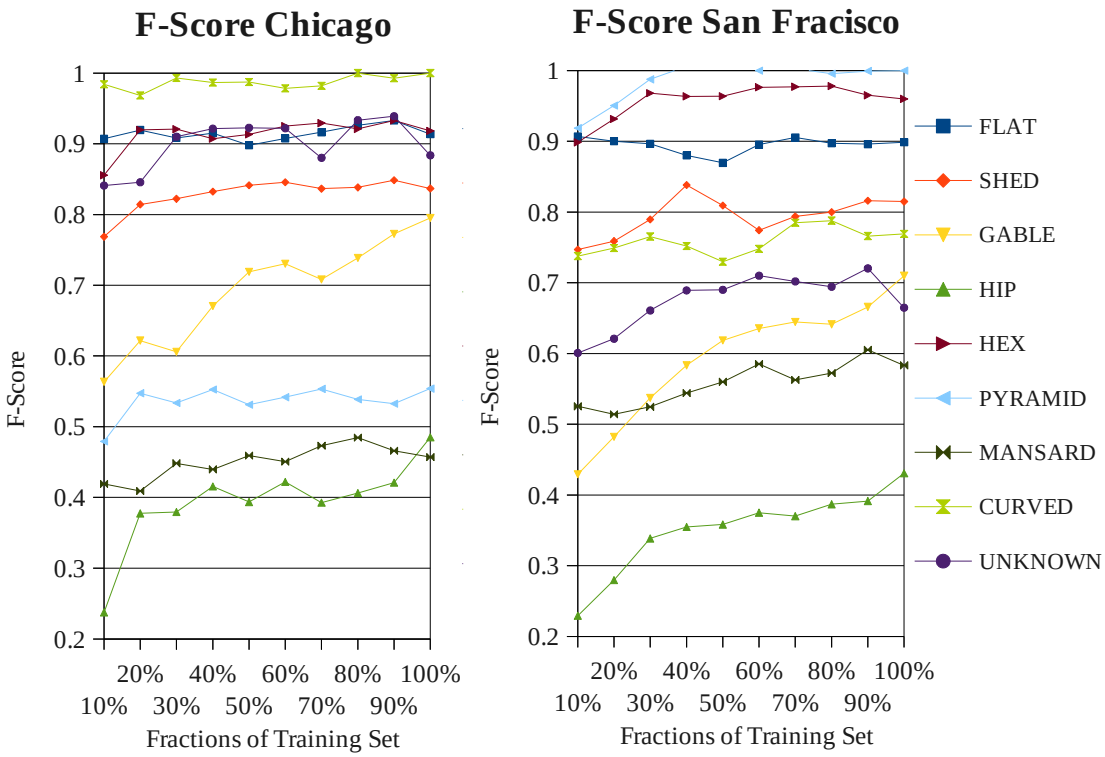


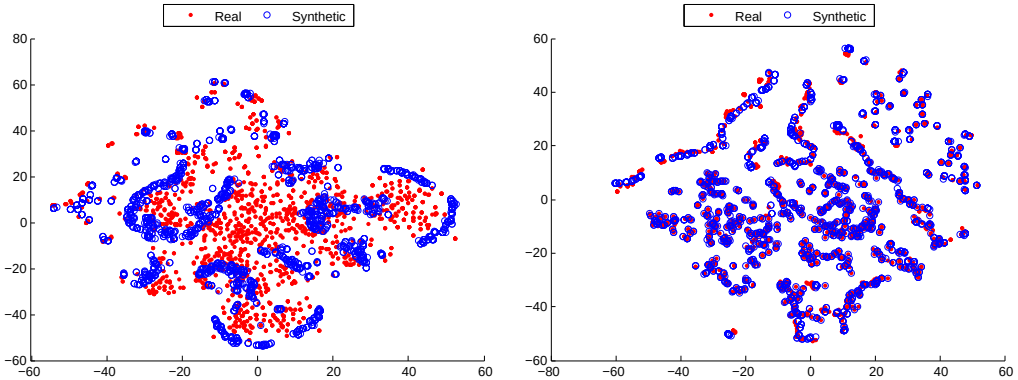
Figure 3.16. The F-Score of each roof style obtained by using an increasing proportion of the training data.

CHAPTER 4

ELIMINATION OF SYNTHETIC GAP

4.1 Introduction

Learning a classifier from synthetic data is unfortunately extremely challenging due to the following reasons. Firstly, the feature distribution of synthetic data generated will shift away from that of real data. Such distribution shift is termed synthetic gap and illustrated in Fig 4.1. The synthetic gap is a major obstacle in using synthetic data to help learning classifiers, since synthetic data may fail to simulate the potential useful patterns of real data for training classifiers. To my knowledge, this synthetic gap problem has never been formally identified nor addressed in the literature. Secondly, since practically a small amount of labeled images may be available, it is necessary to jointly learn from synthetic and real data. The learning process must be automatically leveraged between synthetic data and real data.



(a) (b)

Figure 4.1. t-SNE visulization of synthetic gap using the data from SRC dataset. (a) synthetic gap of real and synthetic data; (b) MCAE bridges the synthetic gap.

To better learn a classifier from synthetic data, I propose a novel framework – Multichannel Autoencoder (MCAE) which is an extension of sparse autoencoder. The training step of MCAE is a process of bridging the synthetic gap between the real and the synthetic data by learning the mapping from (1) synthetic to real data and (2) real to real

data. Critically, such mapping try to keep the real data while enforce MCAE to learn a transfer from the synthetic data to the real data. I thus can generate more synthetic data which will simulate the real data when the learned mapping is applied to them.

4.2 Related Work

A large number of domain adaptation methods have been proposed over the recent years, and here I focus on the most related ones. Multiple methods perform unsupervised domain adaptation by matching the feature distributions in the source and the target domains. Some approaches perform this by reweighing or selecting samples from the source domain [61] [62] [63], while others seek an explicit feature space transformation that would map source distribution into the target ones [64] [65] [66]. An important aspect of the distribution matching approach is the way the (dis)similarity between distributions is measured. Here, one popular choice is matching the distribution means in the kernel-reproducing Hilbert space [61] [62], whereas [67] [68] map the principal axes associated with each of the distributions. my approach also attempts to match feature space distributions, however this is accomplished by modifying the feature representation itself rather than by reweighing or geometric transformation. Also, my method uses (implicitly) a rather different way to measure the disparity between distributions based on their separability by a deep discriminatively-trained classifier.

Several approaches perform gradual transition from the source to the target domain [65] [67] by a gradual change of the training distribution. Among these methods, [69] does this in a deep way by the layerwise training of a sequence of deep autoencoders, while gradually replacing source-domain samples with target-domain samples. This improves over a similar approach of (Glorot et al., 2011) that simply trains a single deep autoencoder for both domains. In both approaches, the actual classifier/predictor is learned in a separate step using the feature representation learned by autoencoder(s). In contrast to [70] [69], my approach performs feature learning, domain adaptation and classifier learning jointly, in a

unified architecture, and using a single learning algorithm (backpropagation). I therefore argue that my approach is simpler (both conceptually and in terms of its implementation). my method also achieves considerably better results on the popular OFFICE benchmark. While the above approaches perform unsupervised domain adaptation, there are approaches that perform supervised domain adaptation by exploiting labeled data from the target domain. In the context of deep feed-forward architectures, such data can be used to fine-tune the network trained on the source domain [71] [72] [73]. Finally, a recent and concurrent report by [74] also focuses on domain adaptation in feed-forward networks. Their set of techniques measures and minimizes the distance of the data means across domains. This approach may be regarded as a first-order approximation to my approach, which seeks a tighter alignment between distributions.

Autoencoder is one type of neural network and its output vectors have the same dimensionality as the input vectors [75]. The hidden representation obtained by training a sparse autoencoder followed by a parameters fine tuning is useful in pre-training a deeper neural network. Recently autoencoder with its different variants [76, 77] also exhibit the success in learning and transferring sharing knowledge among data source from different domains [78, 79, 80], thus benefit other machine learning tasks.

Transfer Learning aims to extract the knowledge from one or more source tasks and applies the knowledge to a target task. Transfer learning has been found helpful in many real world problems, such as in sentiment classification [81], web page classification [82] and zero-shot classification of image and video data [83, 84, 85, 86, 87, 88, 89, 90, 91]. Transfer learning is categorized to three classes [92]: inductive transfer learning, transductive transfer learning and unsupervised transfer learning. The work in this paper falls into a framework of domain adaptation [93, 94] in the transductive transfer learning. Nonetheless, different from previous domain adaptation tasks of different source and target domains, the synthetic gap is caused by the shifted feature distribution of synthetic data

from real data. To solve this problem, my MCAE is developed from the idea of autoencoder.

4.3 Multichannel Autoencoder (MCAE)

In this section, I introduce the MCAE model as illustrated in Fig. 4.2. It can (1) bridge synthetic gap by minimizing the discrepancy between real and synthetic data; and (2) preserve and emphasize the potential useful patterns existed in both real and synthetic data in order to generate the better feature representations used for learning classifiers.

Essentially, synthetic and real data should have similar patterns, a natural idea of bridging synthetic gap is to learning a mapping from the synthetic data to the real data using an autoencoder, and vice versa. MCAE, hence, provides a more flexible way to learn this mapping due to the specific structure of the MCAE. There are two channels in MCAE, left one and right one. Each channel basically is an SAE, however, two channels share the same hidden layer. With this structure, MCAE basically learns two tasks in the same time. By setting different types of input and out data such as the one in denoising autoencoder [95], MCAE is capable for many applications. In my work, to bridge the gap between synthetic data and real data, I set the task in left channel as one that takes synthetic data as input and real data as *reconstruction target*, while the task in right channel use real data in both input and *reconstrution target*. This configuration actually is essentially meaningful that by keeping the *reconstruction target* identical in two channels, MCAE attempts to transform inputs in two channels towards the same target, thus minimize the discrepancy between two input dataset which are synthetic data and real data in my work.

4.3.1 Problem setup. My MCAE is built on the sparse autoencoder (SAE). A basic autoencoder is a fully connected neural network with one hidden layer and can be decomposed into two parts: an encoding and a decoding process. Assume an input dataset with n instances $X = \{x_i\}_{i=1}^{n}$ where $x_i \in \mathbb{R}^m$ and m is the dimension of each instance. Encoding typically transforms input data to hidden layer representation using an affine mapping

squashed by a sigmoid function:

$$h_e(x_i) = f(W_e x_i + b_e) \tag{4.1}$$

where $f(\cdot)$ is a sigmoid function and $\theta_e = \{W_e, b_e\}$, $W_e \in \mathbb{R}^{k \times m}, b_e \in \mathbb{R}^k$ is a set of unknown parameters in encoding with k nodes in hidden layer.

While in decoding, with parameters $\theta_d = \{W_d, b_d\}$, $W_d \in \mathbb{R}^{m \times k}, b_d \in \mathbb{R}^m$, autoencoder attempts to reconstruct the input data at the output layer by imposing another affine mapping followed by nonlinearity to hidden representation $h_e(x_i)$:

$$h_d(x_i) = f(W_d h_e(x_i) + b_d) \tag{4.2}$$

In above equation $h_d(x_i)$ is viewed as a reconstruction of input x_i. Normally, I impose $h_d(x_i) \approx x_i$. Here x_i play a role of *reconstruction target* in this expression and I use notation $\langle \mathfrak{i}{:}X_i, \mathfrak{t}{:}X_i \rangle$ to denote the configuration of input data short for $\mathfrak{i}$ and *reconstruction target* short for $\mathfrak{t}$ in an autoencoder. X_s and X_r indicate synthetic and real data respectively. By minimizing the reconstruction errors of all data instances, I have following objective function:

$$J(\theta_e, \theta_d) = \frac{1}{n} \sum_{i=1}^{n} (h_d(x_i) - x_i)^2 + \lambda W \tag{4.3}$$

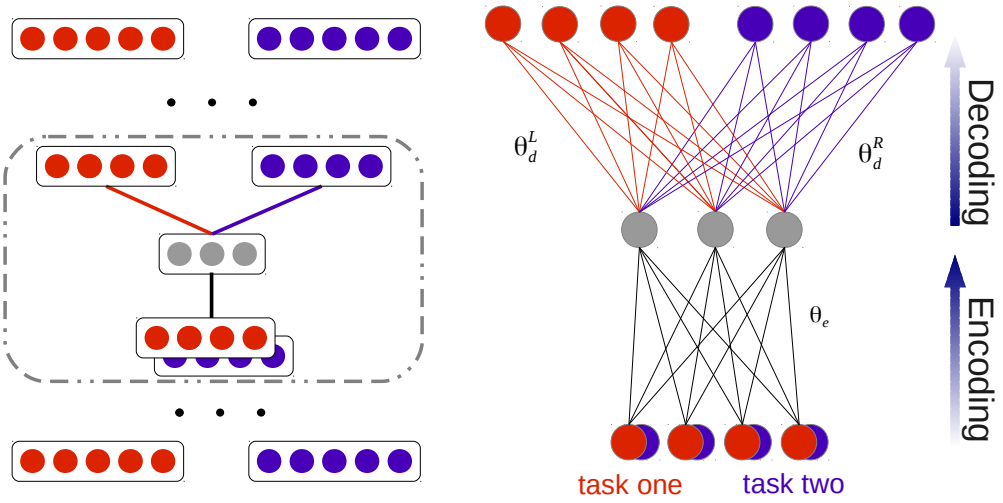
where $W = (\sum W_e^2 + \sum W_d^2)/2$ is a weight decay term added to improve generalization of the autoencoder and λ leverages the importance of this term.

To avoid learning identity mapping in autoencoder, a regularization term $\Theta = \sum_{i=1}^{k} \delta \log \frac{\delta}{\hat{\delta}_i} + (1-\delta) \log \frac{1-\delta}{1-\hat{\delta}_i}$ that penalizes over-activation of the nodes in the hidden layer is added. δ is a sparsity parameter and is set by users and $\hat{\delta}_i = \frac{1}{k} \sum_{i=1}^{k} h_e(x_i)$.

$$J(\theta_e, \theta_d) = \frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i)^2 + \lambda W + \rho\Theta \tag{4.4}$$

ρ controls sparsity of representation in hidden layer.

Note that directly applying sparse autoencoder to my problem does not work well. For example, I can train an autoencoder purely by placing synthetic data in input layer and real data in output layer denoted as $\langle \text{i}{:}X_s,\ \text{t}{:}X_r \rangle$ which however can not bridge the synthetic gap in my problem. Such way of reconstruction is only to complement the missing information in synthetic data from real data but not vice versa



(a) (b)

Figure 4.2. (a) Illustration of the proposed MCAE model in a stacked autoencoder structure, where black edge between two layers are linked to and shared by two tasks, red and blue links are separately connected to left and right task respectively. (b) A zoom in structure of MCAE.

A better representation should be reconstructed by using the information from both real and synthetic data simultaneously. Specifically, I aim at two tasks: one is $\langle \text{i}{:}X_s, \text{t}{:}X_r \rangle^L$ which reconstructs synthetic data towards real data, and the other one is $\langle \text{i}{:}X_r,\ \text{t}{:}X_r \rangle^R$ which uses identical real data for input and *reconstruction target*, where $\langle\cdot\rangle^L$ and $\langle\cdot\rangle^R$ indicate the left and right channel of MCAE.

4.3.2 MCAE model. I propose a multichannel autoencoder that uses a balance regularization to leverage the learning between two tasks, i.e. $\langle \text{i}{:}X_s,\ \text{t}{:}X_r \rangle^L$ and $\langle \text{i}{:}X_r,\ \text{t}{:}X_r \rangle^R$. The structure of this new autoencoder is shown in Fig. 4.2. In this new structure, tasks of two channels will share the same parameters θ_e in encoding process which will enforce autoencoder to reconstruct common structure in both tasks. However, in decoding process, I divide autoencoder to two separate channels that two tasks will have their own parameters θ_d^L and θ_d^R. Dividing autoencoder to two channels at decoding layer enable a more flexible control between the two tasks. Thus autoencoder better leverage the common knowledge from the two tasks.

With two channels in the MCAE, I target to minimize the reconstruction error of two tasks together while taking into account the balance between two channels. The new objective function of the MCAE is given in the following:

$$E = J^L(\theta_e, \theta_d^L) + J^R(\theta_e, \theta_d^R) + \gamma\Psi \tag{4.5}$$

where

$$\Psi = \frac{1}{2}(J^L(\theta_e, \theta_d^L) - J^R(\theta_e, \theta_d^R))^2 \tag{4.6}$$

is a regularization added to balance the learning rate between two channels. This regularization will have two effects on the MCAE. First, Ψ accelerates the speed of optimizing Eq. 4.7, since minimizing Ψ requires both $J^L(\theta_e, \theta_d^L)$ and $J^R(\theta_e, \theta_d^R)$ are small which in turn cause E decreases faster. Second, Ψ penalizes a situation more when difference of learning error between two channels are large, so as to avoid imbalanced learning between two channels.

The minimization of Eq. 4.7 is achieved by back propagation and stochastic gradient descent using Quasi-Newton method. Since the regularization term is added to leverage the balance of different tasks, I have to compute the gradient of parameters θ_e and θ_d^L, θ_d^R

in MCAE.

With two branches in the MCAE, I target to minimize the reconstruction error of two tasks together while taking into account the balance between two branches. The new objective function of the YMAE is given in the following:

$$E = J^L(\theta_e, \theta_d^L) + J^R(\theta_e, \theta_d^R) + \gamma\Psi \tag{4.7}$$

where

$$\Psi = \frac{1}{2}(J^L(\theta_e, \theta_d^L) - J^R(\theta_e, \theta_d^R))^2 \tag{4.8}$$

is a regularization added to balance the learning rate between two branches. This regularization will have two effects on the YMAE. First, Ψ accelerates the speed of optimizing Eq. 4.7, since minimizing Ψ requires both $J^L(\theta_e, \theta_d^L)$ and $J^R(\theta_e, \theta_d^R)$ are small which in turn cause E decrease faster. Second, Ψ penalize a situation more when difference of learning error between two branches are large, so as to avoid imbalanced learning between two branches.

The minimization of Eq. 4.7 is achieved by back propagation and stochastic gradient descent using Quasi-Newton method. In the MCAE, with balance regularization added to the objective, the only difference as opposed to sparse autoencoder is the gradient computation of unknown parameters θ_e and θ_d^L, θ_d^R. I clarify these differences in the following equations:

$$\begin{aligned}
\nabla_{W_e} E =& \frac{\partial J^L}{\partial W_e} + \frac{\partial J^R}{\partial W_e} + \gamma(J^L - J^R)(\frac{\partial J^L}{\partial W_e} - \frac{\partial J^R}{\partial W_e}) \\
\nabla_{b_e} E =& \frac{\partial J^L}{\partial b_e} + \frac{\partial J^R}{\partial b_e} + \gamma(J^L - J^R)(\frac{\partial J^L}{\partial b_e} - \frac{\partial J^R}{\partial b_e})
\end{aligned} \tag{4.9}$$

and

$$
\begin{aligned}
\nabla_{W_d^L} E &= \frac{\partial J^L}{\partial W_d^L} + \gamma(J^L - J^R)\frac{\partial J^L}{\partial W_d^L} \\
\nabla_{b_d^L} E &= \frac{\partial J^L}{\partial b_d^L} + \gamma(J^L - J^R)\frac{\partial J^L}{\partial b_d^L} \\
\nabla_{W_d^R} E &= \frac{\partial J^R}{\partial W_d^R} + \gamma(J^L - J^R)(-\frac{\partial J^R}{\partial W_d^R}) \\
\nabla_{b_d^R} E &= \frac{\partial J^R}{\partial b_d^R} + \gamma(J^L - J^R)(-\frac{\partial J^R}{\partial b_d^R})
\end{aligned}
\tag{4.10}
$$

The exact form of gradients of θ_e and θ_d^L, θ_d^R varies according to different sparsity regularization Θ used in the framework.

4.3.3 The advantages of MCAE over alternative Configurations. My MCAE enforces autoencoder to learn useful class patterns from the two tasks simultaneously. Thus it helps with capturing a common structure of synthetic and real images. Another alternative way is to concatenate the input and target of the two tasks $\langle \mathfrak{i}{:}X_s X_r,\ \mathfrak{t}{:}X_r X_r \rangle$ for autoencoder. I annotate the usage of this autoencoder as Concatenate-Input Autoencoder (CIAE), since this autoencoder learns concatenated tasks at the same time. Such configurations however may result in an unbalanced optimization for these two tasks: the optimization process of one task will take over the process of the other one. It results in a biased reconstructed hidden layer of the autoencoder and thus a limited classification performance.

4.4 Evaluation

The proposed MACE has been evaluated on two datasets I introduced in Chapter 2. The first dataset is Satellite Roof Classification (SRC) and the second dataset is handwritten digit dataset from UCI machine learning repository.

Synthetic data are created to highlight the potential useful pattern existed in real images. I have two stages of generating synthetic data. In the first stage, for each real

data used to train MCAE, a synthetic version that best matching appearance of the real data is generated; thus pairs of corresponding real and synthetic data can be used to train the MCAE. In the second stage, more synthetic data could be derived using synthetic data generated in the first stage by both interpolation and extrapolation. To distinguish the set of synthetic data used in these two stages, I use abbreviation *Syn I* and *Syn II* to represent them respectively.

4.4.1 Experiment Settings. I fix the configuration of MCAE as $\langle \mathfrak{i}{:}X_s,\ \mathfrak{t}{:}X_r \rangle^L$ (left channel) and $\langle \mathfrak{i}{:}X_r,\ \mathfrak{t}{:}X_r \rangle^R$ (right channel). Specifically, the left channel is the reconstruction process from synthetic data to real data, while the right channel works in the same way as a standard SAE. my experimental results will show that the representations learned in such way greatly benefit the performance of classifiers I compared.

In the experiments two different classifiers of utilizing learned representations from MCAE (from Sec. 3) are compared. In the first scenario, MCAE encodes input data to a representation (feature) in the hidden layer and a SVM using RBF kernel is employed in this case to show the performance of the classification. In the second scenario, MCAE takes the input images and produces the reconstructed images at the output layer. Features, in this case, are images, therefore can be fed to Convolutional Neural Network (CNN) for classification. In my experiments I build a LeNet-5 [12] which is originally created for digit recognition. I show that using the same number of input data, the performance of the CNN prefers to the data produced by the MCAE.

I summarize all evaluations and comparisons using F-1 score, which is defined as:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.11}$$

4.4.2 Result. To better evaluate the performance of the proposed MCAE, I compare MCAE with Concatenate-Input Autoencoder (CIAE) [96] and Sparse Autoencoder (SAE)

[95]. In these experiments, I evaluate the performance on two classifiers: a CNN using reconstructed images and SVM using encoded hidden layer representation. I present the results of these comparisons in Table 4.1 and Table 4.2 for SRC and handwritten digit datasets respectively. It could be observed from these two tables that although the performance of the CIAE is close to MCAE, the proposed MCAE gets a better performance almost in all the comparisons.

Table 4.1. F1-score of roof style classification using reconstructed images (in CNN) and encoded image features (in SVM). Second column shows the data used to train the autoencoder in the first column. In classification, Real+*Syn II* are used in the training of CNN and SVM. ; *Syn I* + Real means that I use concatenation of the Syn I and real images as the input for the corresponding autoencoders.

	Data to train autoencoder	CNN Reconstructed	SVM Encoded
MCAE	$\langle$i:*Syn I*, t:Real$\rangle^L$ $\langle$i:Real, t:Real$\rangle^R$	**0.68**	**0.80**
CIAE	$\langle$i:*Syn I* + Real, t:*Syn I* + Real$\rangle$	0.68	0.78
SAE	$\langle$i:*Syn I*, t:*Syn I*$\rangle$	0.63	0.59
SAE	$\langle$i:Real, t:Real$\rangle$	0.62	0.62

Table 4.2. F1-score of handwritten digit recognition.

	Data to train autoencoder	CNN Reconstructed	SVM Encoded
MCAE	$\langle$i:*Syn I*, t:Real$\rangle^L$ $\langle$i:Real, t:Real$\rangle^R$	**0.98**	**0.96**
CIAE	$\langle$i:*Syn I* + Real, t:*Syn I* + Real$\rangle$	0.97	0.96
SAE	$\langle$i:*Syn I*, t:*Syn I*$\rangle$	0.94	0.91
SAE	$\langle$i:Real, t:Real$\rangle$	0.95	0.65

To qualitatively show actual images and synthetic images do look more similar after being processed by MCAE, examples of roof style images are shown in Figure 4.3.

Synthetic data help learning a better classifier. I designed another group of experiments. In these experiments three different configurations of data are either reconstructed and encoded using the proposed MCAE, then used to train a CNN or a SVM in the experiments. All results from these experiments are compared in Table 4.3 and Table 4.4 respectively. An interesting thing to notice is that in experiments, using synthetic data can

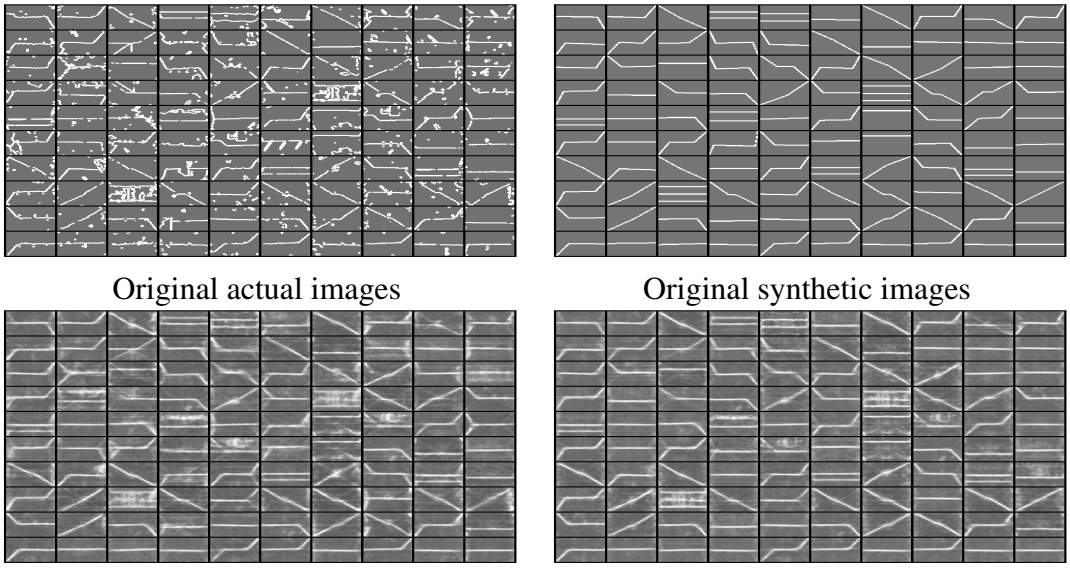


Figure 4.3. Examples of images before and after being reconstructed by MCAE. It could be observed from the images that actual images and synthetic images look much more similar after being processed by MCAE.

only achieve the same result as using a combination of real and synthetic data. This result proves that the distribution of the real data in this case is almost overlapping with the distribution of the synthetic data.

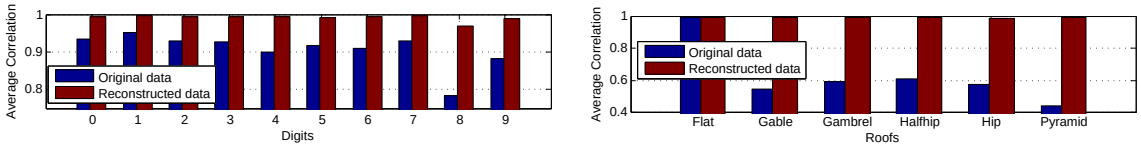
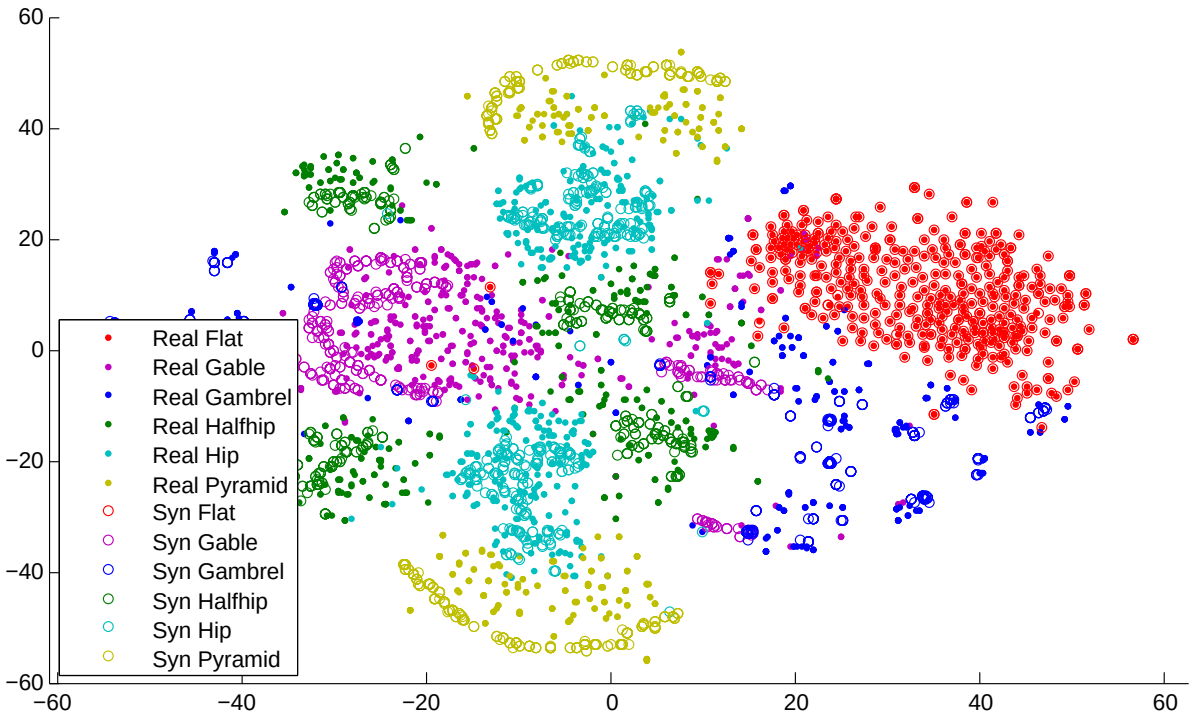


Figure 4.4. Correlation between real and corresponding best matching *Syn I* data.

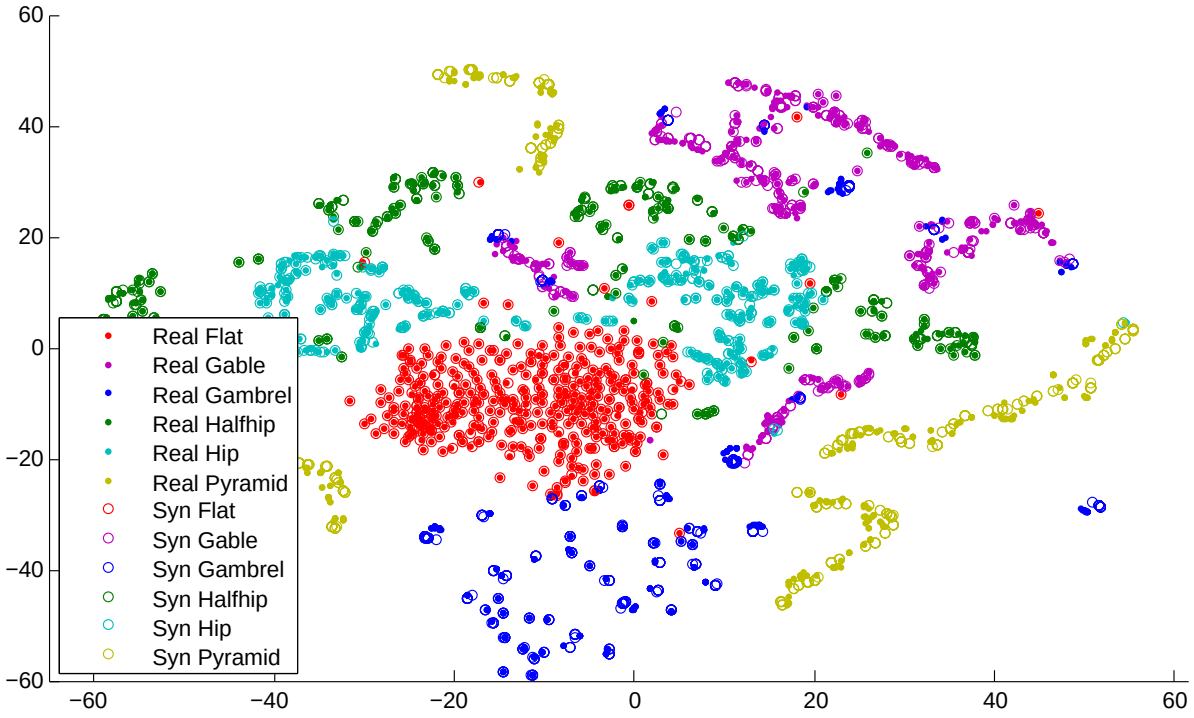
MCAE bridges the synthetic gap. I compare the correlation defined as:

$$\text{Corr} = \frac{\text{Cov}(X, Y)}{\text{Var}(X)\text{Var}(Y)} \tag{4.12}$$

between real and *Syn I* data before and after being reconstructed by the MCAE. The intention of these comparisons is to show that real synthetic images become much more alike each other in terms of the appearance after being reconstructed by the MCAE. The results



(a)



(b)

Figure 4.5. t-SNE [4] visualization of synthetic gap bridged by MCAE. (a) Data distributions of each class of SRC dataset. For many data instances, the (circle) real and (dot points) synthetic data are not overlapping. This is synthetic gap. (b) Data distributions of the reconstructed images by MCAE for each class of SRC dataset. The reconstructed images of all the real (circle) and synthetic (dot points) are almost overlapped. It means that my MCAE can bridge the synthetic gap.

Table 4.3. F1-score of roof style classification by classifier (CNN and SVM) using different set of data reconstructed of encoded using the proposed MCAE.

	Feature type	Real	*Syn II*	Real+*Syn II*
CNN	Reconstructed	0.65	0.68	0.68
SVM	Encoded	0.77	0.78	0.80

Table 4.4. F1-score of handwritten digit recognition.

	Feature type	Real	*Syn II*	Real+*Syn II*
CNN	Reconstructed	0.94	0.96	0.96
SVM	Encoded	0.96	0.96	0.98

are shown in Fig. 4.4. It is shown that my method almost achieves 100% correlation between real and *Syn I* when both data are reconstructed by the proposed MCAE. That means the proposed MCAE bridges the synthetic gap between the real data and the synthetic data. The results are shown in Fig. 4.5. It intuitively shows that my MCAE can help bridge the synthetic gap between real and synthetic data.

CHAPTER 5

CREATION OF SYNTHETIC DATA IN FEATURE SPACE

5.1 Introduction

Synthesis of data in data space usually is limited to specific problems, and generation of the synthetic data requires a through understanding of data. However, for vision task, data representation in data space, a image, usually has very high dimension. Unless data represent some specific patterns and there is way to effectively reduce the dimension, it is often very hard to implement such techniques in most of problems. Actually, it is not possible to model data using a parametric model, as introduced in previous chapter, in most of real problems.

A better but harder space to do data synthesis is in feature space. It is much more interesting if data synthesis could be done in feature space, because it is a space that is general to almost every problems. if synthesis succeeds, the method is no longer restricted to specific problem and thus can applied to all problems. However, it is never easy to insert synthetic data in feature space. Because on the one hand, in feature space all feature vectors basically are represented just a high dimensional abstract point without knowing any of its actual representation in data space. On the other hand, a good representation of data in feature space highly depends on actual features extracted. A good representation in feature space should maximize both the dissimilarity between classes and coherence among data within a class.

5.2 Related Work

Generate more data in feature space has been long studied in imbalanced learning problem [97], where data are created for minority class in learning process to balance the amount of data in different classes in order to rectify the bias of learning that towards majority classes.

The simplest way to generate more data is by replication. Several previous researches [98] and [99] have discussed this strategy in their work. They noted that oversampling with replacement does not significantly improve the recognition accuracy on minority class. Such phenomenon can be interpreted as a split of decision region. As the minority class is over-sampled by increasing amounts, the decision region for the minority class becomes very specific and will cause new splits for decision region [97]. Learning method in this case will learn these decision regions specifically which leads to an overfitting.

instead of simply duplicating data, several methods are invented in last decade in which new data are generated to increase data coverage in feature space using linear combination of existing data. It is worth mentioning that the synthetic minority oversampling technique (SMOTE) is a powerful method that has shown a great deal of success in imbalance learning. The SMOTE algorithm creates artificial data based on the feature space similarities between existing minority examples. Specifically, for each data in minority class, a new data is generated as a interpolation between local data and one of its randomly chosen k-nearest neighbors. Thus using method, the resulting synthetic instance is a point along the line segment joining local data and its neighbor, shown in Figure 5.1. As a result of interpolation, larger and less specific regions are learned, thus, learner paying attention to minority class samples without causing overfitting.

Despite the success SMOTE achieved, the drawback of it is also very evident, including over generalization and variance. Specifically, SMOTE generates the same number of synthetic data samples for each original minority example without considering neighborhood. To this end, various adaptive sampling method rooting on SMOTE have been proposed to overcome the limitations.

One of representatives of these method is Borderline-SMOTE algorithm in [100]. Extended from basic SMOTE, Bordline-SMOTE method focus on samples that are near

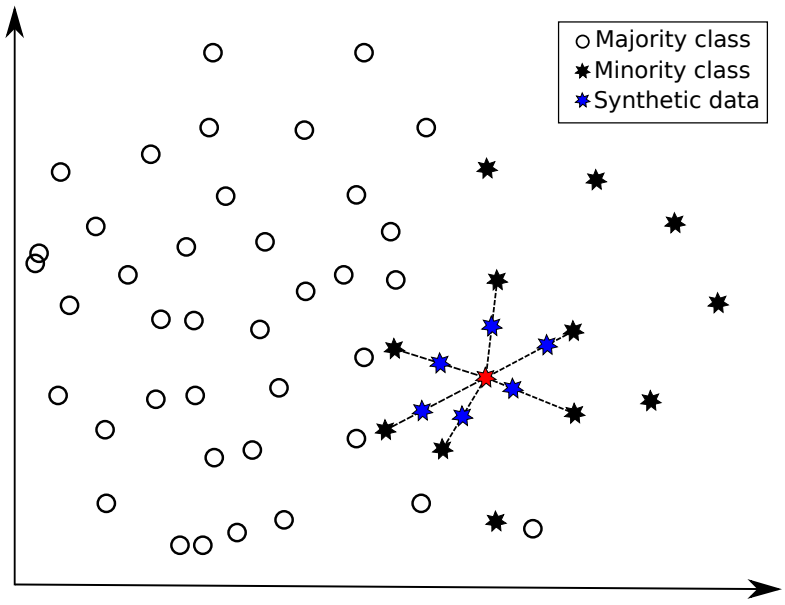


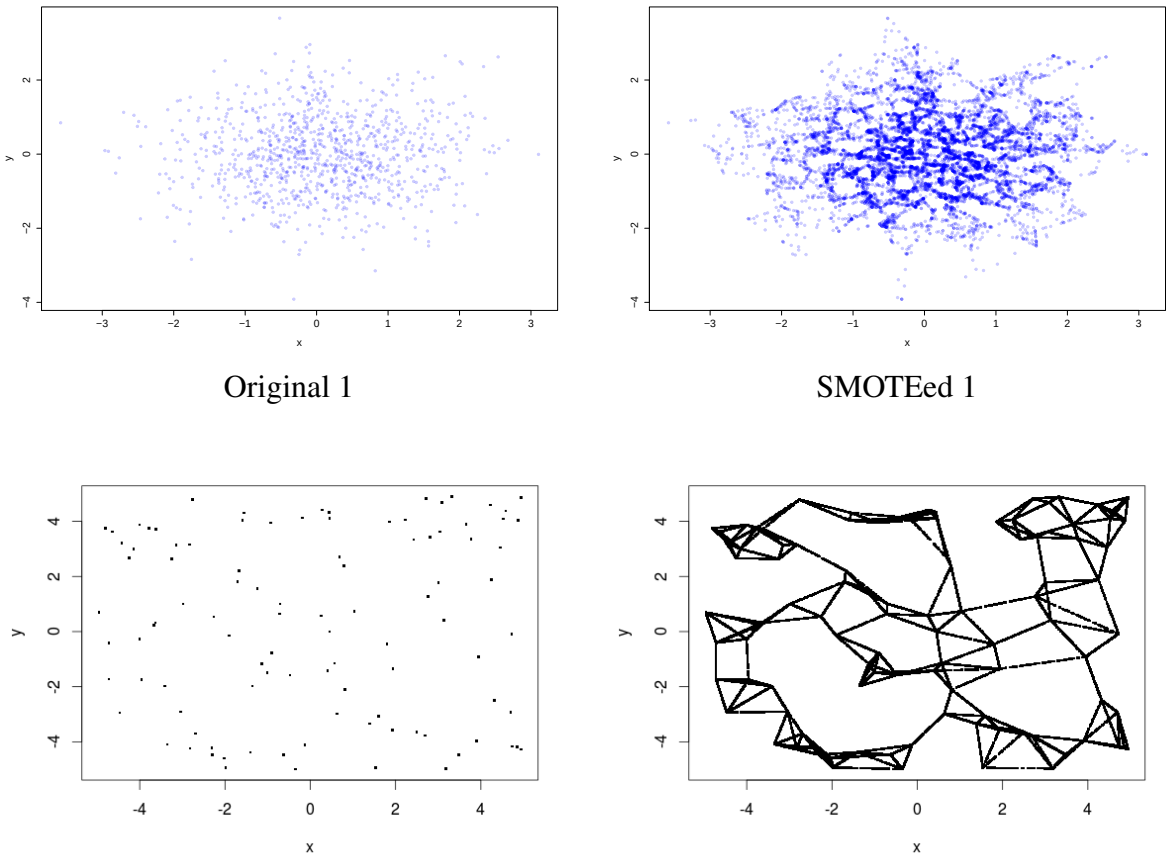
Figure 5.1. The F-Score of each roof style obtained by using an increasing proportion of the training data.

the class border and synthesize new data only for these samples. Of particular interest with this algorithm is the techniques used to identify borderline samples. To do so, Borderline-SMOTE actually look into the ratio of different class samples in K-nearest neighbors of a borderline sample. Thresholds are set to distinguish among outliers, non-borderline samples and borderline samples. Using different seed selection strategies and learning method, several similar methods based on borderline-SMOTE are proposed later these years [101][102].

Researchers combine boosting and SMOTE in [103] that SMOTE is carried out in each round of boosting to enable each learner to be able to sample more of the minority class cases so as to learn better and broader decision regions for the minority class. They also conjecture that introducing the SMOTE procedure also increases the diversity amongst the classifies in the ensemble, as in each iteration the approach will produce a different set of synthetic examples and therefore different classifiers.

I often use SMOTE and its variants in my work too. Most of time I do benefit

by using it. However, occasionally there are problems when amount of data in different classes are extremely skewed, for example 100:1. Using SMOTE, in this case, to matching minority data number with majority classes will cause locally aggregation of synthetic data thus destroy inherent structure of original data. Examples are shown in Figure 5.2. It could be observed from the figure that due to SMOTE always insert new synthetic data along a line segments linking local point and its neighbors. The density of points on the line segments becomes extremely high when SMOTE are intensively applied on the dataset, thus cause severe data aggregation locally around each data and its nearest neighbors.



Original 1 SMOTEed 1

Original 2 SMOTEed 2

Figure 5.2. Examples of applying SMOTE on two dataset. Original datasets are shown on left column, results of using SMOTE are shown on the right. A dataset with samples following a normal distribution is shown in first row. In the second row, dataset follows a uniform distribution.

Another method combining boost but using different data synthesis strategy is DataBoost IM approach proposed by [104]. In the DataBoost-IM method, hard examples from both the majority and minority classes are identified druing execution of the boosting algorithm. subsequently, the hard examples are used to separately generate synthetic examples for the majority and minority classes. Specifically, they first separately identify hard examples from and generate synthetic examples for the minority as well as the majority classes. Secondly, they generate synthetic examples with bias information toward the hard examples on which the next component classifier in the boosting procedures needs to focus. Thirdly, the class frequencies in the new training set are rebalanced to alleviate the learning algorithm's bias toward the majority class. Fourthly, the total weights of the different classes in the new training set are rebalanced to force the boosting algorithm to focus on not only the hard examples, but also the minority class examples. In this way, they could improve the prediction of both the minority and majority classes. Rather than data generation using interpolation in SMOTE based method, the data generation in DataBoost-IM is based on a distribution of seed example. New data are drawn as samples from a normal distribution with mean and variance computed based on seed examples. Issues with data generation like this are that first, an assumption of normal distribution for seed examples is too strong that the number of seed examples may be too less to form any distribution. Second, there may exist several sub-parts in seed examples, which is inaccurate if Gaussian distribution with only one component is assumed.

5.3 The Proposed Method

Although existing feature space data synthesis algorithms show satisfactory results in some researches. All of these methods do not consider neither global distribution nor local structure of data. All behaviors of data synthesis are done either randomly sampling from a assumed distribution or by simple interpolation between data, which results in a dataset that could be totally distinct from the original one when a great number of synthesis

operation is carried out. Thus the classification result will bias to data that are more similar to training data and failed to recognize data that are moderate vary from the training data. The reason is because existing methods do not treat entire data as a whole and do not investigate the relationship among the samples. Each sample is recognized just as a independent sample in existing method. Existing method basically do not admit the duty of each sample in dataset.

In my work, I propose a novel algorithm that synthesize data in feature space by considering the structure or distribution of entire dataset. Suppose in feature space a dataset named $D = \{d_{\mathrm{D}}^i\}_{i=1}^i$ is given, where each d^i is a feature vector. I consider D as a subset that is sampled from a distribution P_{all} which is the distribution of all possible samples from the sample data source. my goal is to have another dataset labelled as D_s which is also sampled from P and is a superset of D that $D \subset D_s$. So D_s is a dataset that contains not only the original D but also all synthesized samples. At this point, let's think about this procedure in a reverse way. Instead of consider D_s as a set up-sampled from D. I consider D as a dataset that is down-sampled from D_s. Labelling distribution of D_s as P_{Ds} and distribution of D as P_{D}, since D can be considered to be sampled from D_s, they share a similar distribution that P_{D} is very close to P_{Ds}. Since the real P is unknown, I make an assumption that D_s so far is the whole set of the problem. During the procedure of down-sampling, all synthetic data are removed, which will cause a divergence between distribution of D_s and distribution of D. Such divergence is denoted as $Div(P_{\mathrm{Ds}}, P_{\mathrm{D}})$ which stands for a divergence caused by a down-sampling from P_{Ds} to P_{D}.

To this end, if I are able to compute $Div(P_{\mathrm{Ds}}, P_{\mathrm{D}})$, I can find a way to recover distribution of D_s from D. Using recovered distribution as a target, I are able to synthesize data from D to minimize $Div(P_{\mathrm{Ds}}, P_{\mathrm{D}})$. However, $Div(P_{\mathrm{D}}, P_{\mathrm{Ds}})$ is unknown, since D_s is unknown so far. Since I assume that D is the dataset that is sampled from D_s and has similar distribution with D_s, I can actually think of all samples in D as *skeleton samples*

of D_s. Since in life it is often the rare objects that are most interesting. This carries over to this problem too. So *skeleton samples*, D, can be seen as the most important samples with which only D_s can maintain the same distribution. And all synthetic data in D_s and abandoned during down-sampling will cause a divergence $Div(P_{\text{Ds}}, P_{\text{D}})$ which can be considered as a "information loss" of this processing. Since P_{Ds} and P_{D} are similar, the entire down-sampling process is very similar to a low pass filtering process in which the most essential parts (signals with low frequency) of the signal are kept and high frequency parts are removed. I make another assumption that $Div(P_{\text{Ds}}, P_{\text{D}})$ is proportional to the downsampling ratio r_{DsD} from D_s to D. For now, let us suppose this assumption hold. Now, image I can find a dataset denoted as D_d which is down-sampled from D and contains all *skeleton samples* of D. I could figure out why all other samples of D which cause a divergence of $Div(P_{\text{D}}, P_{\text{Dd}})$ get abandoned and what their roles are in maintaining the distribution of D. By learning these information, I are able to apply the information to synthesize D_s from D. Since basically the down-sampling process from D_s to D and the down-sampling process from D to D_d are same, all they result in are *skeleton samples* of the superset. An example of this procedure is shown in Figure

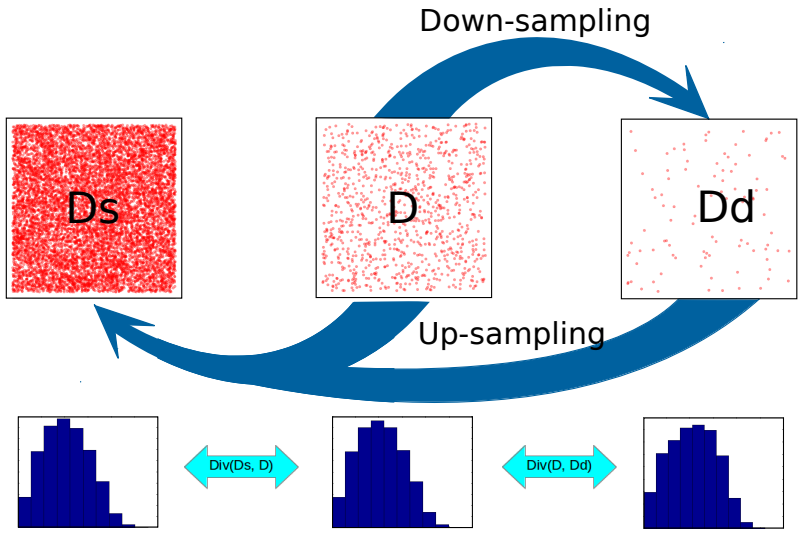


Figure 5.3. An illustration of the proposed method.

So, the proposed method contains two phases. First, I get D_d by down-sampling D. In second phase, by learning information from the down-sampling in the first phase, I could reversely apply all the information to an up-sampling process by synthesizing more data from D and get D_s.

5.3.1 Down-sampling. Given the original dataset $D = \{d_{\text{D}}^i\}_{i=1}^n$, I compute D_d by down-sampling from D. The number of samples getting kept in D_d is given by $|D_d| = (1-r_{\text{DDd}})\times |D|$, where $r_{\text{DDd}}^{\downarrow}$ is the down-sampling ratio. During down-sampling process, an indicate function $I_{\text{D}}(d_i) \in \{0, 1\}$ is used to indicate the sample d_i is removed when $I_{\text{D}}(d_{\text{D}}^i) = 0$ or the sample is kept when $I_{\text{D}}(d_{\text{D}}^i) = 1$.

As I mentioned, the goal of down-sampling procedure by giving a down-sampling ratio r_{DDd} is to find out $|D_d|$ samples that still have very similar distribution P_{Dd} with P_{D}. To represent the distribution for dataset D, a complete connected graph G_{D} is built on D. Then a distance function $\rho_{\text{D}}(d_{\text{D}}^i, d_{\text{D}}^j) \rightarrow \mathfrak{R}$ is defined for every edge connected by d_{D}^i and $(d_{\text{D}}^i, d_{\text{D}}^j) \in D \times D$, where $\mathfrak{R}$ denotes a real number. I require $\rho_{\text{D}}(d_{\text{D}}^i, d_{\text{D}}^j) \geq 0$ for all pairs $(d_{\text{D}}^i, d_{\text{D}}^j) \in D \times D$ and $\rho_{\text{D}}(d_{\text{D}}^i, d_{\text{D}}^j) = 0$ if $d_i = d_j$. Indeed, for all pairs in $D \times D$, the distance function defines a distance matrix M_{D} that $M_{\text{D}}(i, j) = \rho_{\text{D}}(d_{\text{D}}^i, d_{\text{D}}^j)$. At this point, I define the distribution P_{D} as a distribution of all entry values of M_{D}. To do so, I discretize all values of M_{D} to a histogram $H(D) = \{b_{\text{D}}^i\}_{i=1}^k$, where b_{D}^i represents a bin of the histogram and $\sum_{i=1}^k b_{\text{D}}^i = 1$. Since the histogram is indeed decided by dataset, I use a function form $H(D)$ to emphasize the histogram comes from a distance matrix of D.

Similarly, a similar distance matrix M_{Dd} could be defined for D_d with value of each entry defined as $M_{\text{Dd}}(i, j) = I_{\text{D}}(d_{\text{D}}^i) I_{\text{D}}(d_{\text{D}}^j) \rho_{\text{D}}(d_{\text{D}}^i, d_{\text{D}}^j)$. So if any one of sample is removed from edges linked to it, the distance of the edge will be zero. Another histogram $H(D_d)$ is created to save non-zero entries of M_{Dd}. I only save non-zero entries of M_{Dd}, because after down-sampling a lot of entries become zeros, saving these entries making $H(D_d)$ skew to the first bin which saves all distance of value 0. To compute the divergence be-

tween $H(D)$ and $H(D_d)$, there were tons of ways to do that. I have tried Kullback-Leibler divergence and cosine distance so far. By down-sampling, I minimize the divergence $Div(H(D), H(D_d))$ between $H(D)$ and $H(D_d)$. Suppose the divergence is computed by Kullback-Leibler divergence the objective function can be written as:

$$E = Div(H(D), H(D_d)) = \sum_{i}^{k} b^i_{\text{D}} \ln \frac{b^i_{\text{D}}}{b^i_{\text{Dd}}} + b^i_{\text{Dd}} \ln \frac{b^i_{\text{Dd}}}{b^i_{\text{D}}} \tag{5.1}$$

Solving this objective function turns out to be a process that finds $|D_d|$ samples from D whose indicate function $I(d^i_{\text{D}}) = 1, d^i_{\text{D}} \in D$. Since indicate function get integer value of either 0 or 1, the objective is indifferentiable and can not be optimized by gradient based method. So I propose to use an approximate method and obtain D_d using random search algorithm [105]. While optimizing Equation 5.1, I also take into account the local density of datasets that I hope the resulting dataset D_d should be drawn from D according to samples' local density. That means the higher the local density in D, the more samples will be drawn from their and vice versa. Here the local density is defined quite relative to the down-sampling process. The local density is represented by the sum of indicate function of k nearest neighbors. Given a sample $d^i_{\text{D}}, d^i_{\text{D}} \in D$, its k nearest neighbors are represented as $N_{d^i_{\text{D}}}$. For dataset D_d, suppose d^i_{D} is kept by down-sampling process, I will count how many neighbors of it in $N_{d^i_{\text{D}}}$ are kept by down-sampling too, say k^i_{Dd} samples are kept. Since down-sampling has to draw samples according to local density, the ratio between k^i_{Dd} and $N_{d^i_{\text{D}}}$ mush be very close to down-sampling ratio $r^{\downarrow}_{\text{DDd}}$. So, I add another term to objective function to penalize the down-sampling process if local density are not kept, which will give a new objective function as below:

$$E = Div(H(D), H(D_d)) = \beta \sum_{i}^{k} b^i_{\text{D}} \ln \frac{b^i_{\text{D}}}{b^i_{\text{Dd}}} + b^i_{\text{Dd}} \ln \frac{b^i_{\text{Dd}}}{b^i_{\text{D}}} + (1-\beta) \sum_{i}^{n} (\frac{k^i_{\text{Dd}}}{|N_{d^i_{\text{D}}}|} - r^{\downarrow}_{\text{DDd}})^2 \tag{5.2}$$

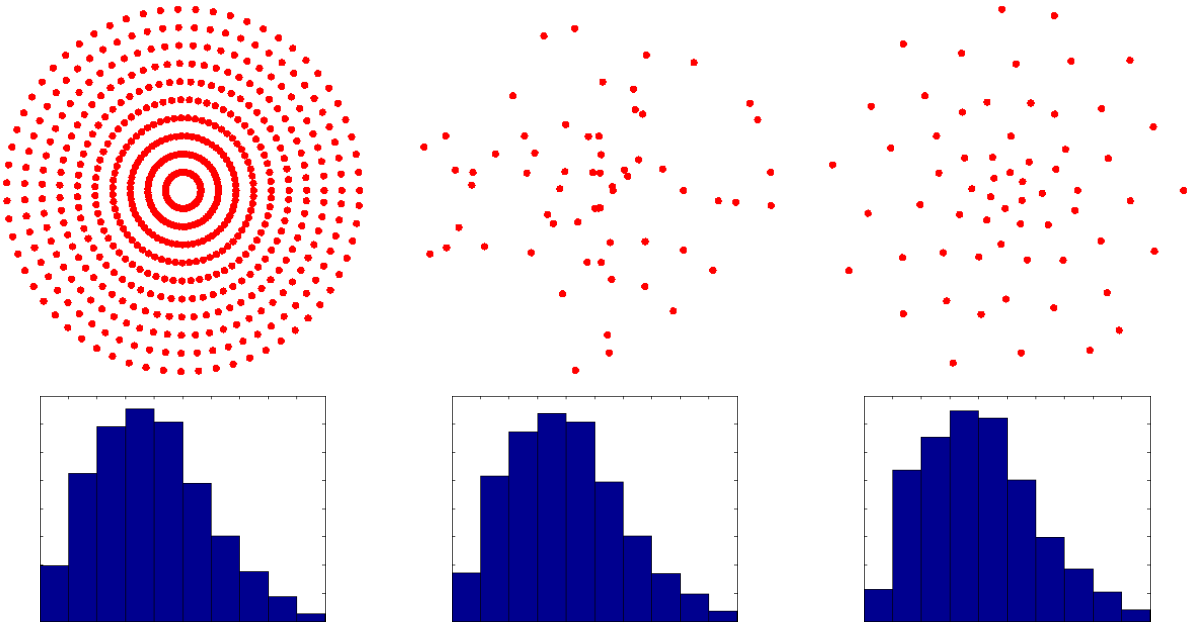


Figure 5.4. Comparison of down-sampling using objective function with and without the 2nd term. Figures in first row show the data, the distribution of which is presented in 2nd row. Dataset is down-sampled to 1/10 of the original size.

An example of down-sampling process is shown in Figure 5.4. The entire optimization is controlled by a pre-set number of iteration and a minimum error change between iterations. The optimization will hit the end when any of these two criteria is met.

5.3.2 Up-sampling. Given an up-sampling ratio $r^{\uparrow}_{\text{DDs}}$, where dataset D is up-sampled to D_s by inserting $(r^{\uparrow}_{\text{DDs}} - 1)|D|$ synthetic samples. Like down-sampling, the essence of up-sampling is to keep global distribution and relative local density consistent between original dataset and up-sampled dataset.

Since target dataset D_s is unknown, I estimate its distribution using the difference of distribution between D and D_d. By using the same convention, suppose the distribution of D_s is represented by a histogram $H(D_s) = \{d^i_{\text{Ds}}\}^k_{i=1}$, I assume following relationship holds:

$$H(D_s) = H(D) + \alpha(H(D) - H(D_d)) \tag{5.3}$$

in which the operations are bin-wise.

Having $H(D_s) = \{d^i_{\text{Ds}}\}_{i=1}^k$ computed using above equation, every synthetic point added is going to decrease the bin-wise difference between $H(D)$ and current $H(D^t_s)$, where t indicate the t-th synthetic data to be added. Suppose I are now adding d^t_{Ds}, I hope by adding this sample, $H(D) - H(D_s)$ could be decreased. That means by adding the sample the distance distribution between the sample and all other existing samples should compensate the difference between $H(D)$ and $H(D_s)$, which is to say the distance distribution d^t_{Ds} produce should be as same as $H(D) - H(D_s)$. One observation is that the distribution produced by newly added sample should be close to distribution of samples that nearby. Such observation could be actually explained by triangle inequality theorem. So to decide location for d^t_{Ds}, I start by searching from current D^t_s for a sample the distance distribution produced by who is closest to $H(D) - H(D_s)$. Then k nearest neighbors of this sample are retrieved. The location of d^t_{Ds} could be decided by interpolating these points. Weights of the interpolation could be solved by least square fit. The up-sampling stops when D_s meet the required size.

CHAPTER 6

CONCLUSION

Synthetic data is extensively used in computer vision area. Although the success of using synthetic data has been shown by many work in the past decades. Usage of synthetic data in more general problems is limited by several factors. First, most work creates and uses synthetic data in a application-specific way that it is hard for other application to benefit. It is interesting if there is an efficient and more generous way to create and use synthetic data, so the method is eligible to most of researches and applications. Second, improvement gained by using synthetic data is limited by a *synthetic gap* existing between actual data and synthetic data. To even boost the performance and unleash the power of synthetic data, such gap should be resolved from many aspects including feature design, learning algorithm design, the way of eliminating the gap and so.

Towards these problems and challenges of using synthetic data. My research has been primarily focusing on solving computer vision related classification and recognition problem using synthetic data. Ways of creating and using synthetic data have been proposed and proved to be effective in my research. To further improve the performance that could be brought by synthetic data, I study the problem of *synthetic gap* and propose many ways to overcome such inherent difference between actual data and synthetic data. To sum up, my doctoral research have following primary contributions.

First, in most of learning based vision research, training data is very valuable and hard to collect, which usually lead to an unsatisfactory machine learning result. Conventional data augmentation method only creates new data which has only small variance from existing ones. In this way, data augmentation fails to introduce many new informations about data and doesn't increase enough data coverage. Thus the performance brought by data augmentation is constraint. In my research, to better simulate actual data, a series of methods that create synthetic data are introduced. A significant problem that data argu-

mentation does not solve and I have to figure out is what makes real data looks like real data and how to simulate unseen data using existing ones. I believe transformation in either data space or parameter space among existing data is the key to this problem. Therefore, methods with new techniques such as: data congealing, registration, parametrization are invented. Different from data argumentation which applies transformation based on single data only, my proposed method learns transformation among data, thus my methods are able to generate synthetic data with larger variance if projected to parameter and feature space which leads to a better simulation of unseen actual data.

Second, directly use synthetic data will most likely cause a failure or unsatisfactory performance in machine learning algorithm. That is mostly because the inherent difference between actual data and synthetic data created using a parametric prototype, which is defined as a *synthetic gap* in my work. In my researches, I eliminate the impact of such difference in two different phases of a machine learning procedure which are in feature extraction and in learning algorithm design. In feature extraction, I designed features which are able to capture the common characteristics of both actual and synthetic data, while ignoring inherent pattern and noises of actual data. In learning algorithm design, learning algorithms are designed to be able to deal with the difference between two kinds of data.

Third, to eliminate the *synthetic gap*, a general framework based on a neural network with novel architecture is proposed. The neural network is trained as a regressor to automatically find out the relationship between two types of data, thus is able to close the gap between two data. With this mechanism synthetic data could better simulate actual data and used in machine learning procedure.

Fourth, create synthetic data in data space and compute sophisticated features from the data is a complicated task. Creating synthetic data in data space narrow down the range of using synthetic data in learning, not only because it requires enough pre-knowledges of problems and data, but also for most problems there is almost no way a synthetic data can

be built in data space to mimic the actual ones. A better solution but much more challenging is to directly create synthetic data in feature space. Inserting new data in feature space is hard. Because data representation in feature space is abstractive that the actual data space representation of a location in feature space is unknown or hard to show. The problem I need to solve here is to figure out answers of two questions: where I are going to insert a new data and why I insert it there? In my research, I propose a general algorithm that create new synthetic data through learning the spatial relation of existing data in feature space. The algorithm first learns distribution and structure from existing data, then each new data is created to preserve these knowledges.

BIBLIOGRAPHY

[1] X. Zhang, A. Zang, G. Agam, and X. Chen, "Learning from synthetic models for roof style classification in point clouds," in *Proceedings of the 22nd ACM SIGSPATIAL international conference on Advances in geographic information systems*. ACM, 2014.

[2] X. Zhang, G. Agam, and X. Chen, "Alignment of 3d building models with satellite images using extended chamfer matching," in *Computer Vision and Pattern Recognition Workshop (CVPRW), 2014 IEEE Computer Society Conference on*. IEEE, 2014.

[3] Ha and H. Bunke, "Off-line, handwritten numeral recognition by perturbation method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 535–539, 1997. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=589216

[4] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-SNE," *Journal of Machine Learning Research*, 2008.

[5] R. Klette, *Concise Computer Vision : An Introduction into Theory and Algorithms*. London: Springer, 2014.

[6] L. Shapiro, *Computer vision*. Upper Saddle River, NJ: Prentice Hall, 2001.

[7] T. Morris, *Computer Vision and Image Processing*. Basingstoke: Palgrave Macmillan, 2004.

[8] B. Jahne, *Computer Vision and Applications : A Guide for Students and Practitioners*. San Diego: Academic Press, 2000.

[9] D. Forsyth, *Computer Vision : A Modern Approach*. Upper Saddle River, N.J. London: Prentice Hall, 2003.

[10] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jul. 1989. [Online]. Available: http://dx.doi.org/10.1016/0893-6080(89)90020-8

[11] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[13] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[14] S. J. Raudys and A. K. Jain, "Small sample size effects in statistical pattern recognition: Recommendations for practitioners," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, no. 3, pp. 252–264, 1991.

[15] H. Kalayeh and D. A. Landgrebe, "Predicting the required number of training samples." *IEEE transactions on pattern analysis and machine intelligence*, vol. 5, no. 6, pp. 664–667, 1983.

[16] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *Information Theory, IEEE Transactions on*, vol. 14, no. 1, pp. 55–63, 1968.

[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and PatternRecognition (CVPR)*, 2012.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[19] O. Chapelle, *Semi-supervised learning*. Cambridge, Mass: MIT Press, 2006.

[20] C. Wang, Z. Li, and L. Zhang, "Mindfinder: Image search by interactive sketching and tagging," in *WWW'10: 19th International World Wide Web Conference, 2010.*, 2010.

[21] X. Zhang, Y. Fu, A. Zang, and G. Agam, "Learning classifiers from synthetic data using a multichannel autoencoder," in *In preparation*, 2015.

[22] A. Zang, X. Zhang, G. Agam, and X. Chen, "Learning based roof style classification in 2d satellite images," in *SPIE. DSS, Geospatial Informatics, Fusion, and Motion Video Analytics V*, 2015.

[23] R. Salakhutdinov, A. Torralba, and J. Tenenbaum, "Learning to share visual appearance for multiclass object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[24] J. A. Richards, *Remote Sensing Digital Image Analysis – An introduction*. Springer Berlin Heidelberg, 2013.

[25] G. M. Weiss, "Mining with rarity: a unifying framework," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7–19, 2004.

[26] T. Varga and H. Bunke, "Effects of training set expansion in handwriting recognition using synthetic data," in *In 11th Conf. of the International Graphonomics Society*. Citeseer, 2003.

[27] ——, "Comparing natural and synthetic training data for off-line cursive handwriting recognition," in *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*. IEEE, 2004, pp. 221–225.

[28] J. Nonnemaker and H. S. Baird, "Using synthetic data safely in classification," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2009.

[29] G. Bal, G. Agam, O. Frieder, and G. Frieder, "Interactive degraded document enhancement and ground truth generation," in *Electronic Imaging 2008*. International Society for Optics and Photonics, 2008.

[30] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[31] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *Advances in intelligent computing*. Springer, 2005, pp. 878–887.

[32] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 1322–1328.

[33] M. Pauly, N. J. Mitra, and L. J. Guibas, "Uncertainty and variability in point cloud surface data," in *Proceedings of the First Eurographics Conference on Point-Based Graphics*, ser. SPBG'04. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 77–84. [Online]. Available: http://dx.doi.org/10.2312/SPBG/SPBG04/077-084

[34] Q.-Y. Zhou and U. Neumann, "Fast and extensible building modeling from airborne lidar data," in *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. ACM, 2008, p. 7.

[35] N. OTSU, "A threshold selection method from gray level histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.

[36] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[37] E. G. Miller, N. E. Matsakis, and P. A. Viola, "Learning from one example through shared densities on transforms," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1. IEEE, 2000, pp. 464–471.

[38] G. Borgefors, "Distance transforms in digital images," in *Computer Vision, Graphics, and Image Processing*, vol. 34. Elsevier, 1986, pp. 344–371.

[39] E. Lipovetsky and N. Dyn, "An efficient algorithm for the computation of the metric average of two intersecting convex polygons, with application to morphing," *Advances in Computational Mathematics*, vol. 26, no. 1-3, pp. 269–282, 2007.

[40] G. Guennebaud and M. Gross, "Algebraic point set surfaces," in *ACM Transactions on Graphics (TOG)*, vol. 26. ACM, 2007, p. 23.

[41] L. Ulanova, Y. Hao, and E. Keogh, "Generating synthetic data to allow learning from a single exemplar per class," in *7th International Conference, SISAP 2014, Los Cabos, Mexico, October 29-31, 2014. Proceedings*, 2014.

[42] T. Obafemi-Ajayi and G. Agam, "Character-based automated human perception quality assessment in document images," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 3, pp. 584–595, 2012. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6062687

[43] X. Zhang and G. Agam, "A learning-based approach for automated quality assessment of computer-rendered images," in *Proc. SPIE, Image Quality and System Performance IX*, 2012.

[44] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Artificial intelligence (IJCAI), 1977 Proceedings of the 5th international joint conference on*, 1977, pp. 659–663.

[45] G. Borgefors, "Hierarchical chamfer matching: a parametric edge matching algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, no. 6, pp. 849–865, 1988.

[46] Q. Zhang, P. Xu, W. Li, Z. Wu, and M. Zhou, "Efficient edge matching using improved hierarchical chamfer matching," in *Circuits and Systems. 2009 IEEE International Symposium on*, 2009, pp. 1645–1648.

[47] J. You, W. Zhu, E. Pissaloux, and H. Cohen, "Hierarchical image matching: a chamfer matching algorithm using interesting points," in *Intelligent Information Systems, 1995 Proceedings of the 3rd Australian and New Zealand Conference on*, 1995, pp. 70–75.

[48] D. M. Gavrila, "Multi-feature hierarchical template matching using distance transforms," in *Pattern Recognition, 1998 Proceedings of 14th International Conference on*, 1998, pp. 439–444.

[49] D. M. Gavrila and V. Philomin, "Real-time object detection for "smart" vehicles," in *Computer Vision, 1999 The Proceedings of the Seventh IEEE International Conference on*, 1999, pp. 87–93.

[50] J. Shotton, A. Blake, and R. Cipolla, "Multiscale categorical object recognition using contour fragment," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 7, pp. 1270–1281, 2008.

[51] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa, "Fast directional chamfer matching," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Computer Society Conference on*, 2010, pp. 1696–1703.

[52] A. Thayananthan, B. Stenger, P. H. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *Computer Vision and Pattern Recognition (CVPR), 2003 IEEE Computer Society Conference on*, 2003, pp. 127–133.

[53] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM*, vol. 13, no. 4, pp. 471–494, 1966.

[54] U. Montanari, "A method for obtaining skeletons using a quasi-euclidean distance," *Journal of the Association for Computing Machinery*, vol. 15, no. 4, pp. 600–624, 1968.

[55] P.-E. Danielsson, "Euclidean distance mapping," in *Computer Graphics and Image Processing*, 1980, pp. 227–248.

[56] P. Felzenszwalb and D. Huttenlocher, "Distance transforms of sampled functions," Cornell Computing and Information Science, Tech. Rep., 2004.

[57] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent point feature histograms for 3d point clouds," *IAS-10*, p. 119, 2008.

[58] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *Graphics, ACM Transactions on*, vol. 21, no. 4, pp. 807–832, 2002.

[59] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 433–449, 1999.

[60] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, Jun. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622407.1622416

[61] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[62] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in neural information processing systems*, 2006, pp. 601–608.

[63] B. Gong, K. Grauman, and F. Sha, "Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 222–230.

[64] S. J. Pan, I. W. Tsang, J. T. Kwok, Q. Yang *et al.*, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[65] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 999–1006.

[66] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 769–776.

[67] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2066–2073.

[68] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2960–2967.

[69] S. Chopra, S. Balakrishnan, and R. Gopalan, "Dlid: Deep learning for domain adaptation by interpolating between domains," in *ICML workshop on challenges in representation learning*, vol. 2, 2013, p. 5.

[70] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 513–520.

[71] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 818–833.

[72] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 1717–1724.

[73] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 584–599.

[74] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[75] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *International Conference on Machine Learning*, 2011.

[76] M. Chen, Z. Xu, K. Q. Weinberger, and Fei, "Marginalized denoising autoencoders for domain adaptation," in *International Conference on Machine Learning*, 2012.

[77] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *International Conference on Machine Learning*, 2011.

[78] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," *Unsupervised and Transfer Learning Challenges in Machine Learning, Volume 7*, p. 43, 2012.

[79] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," *Unsupervised and Transfer Learning Challenges in Machine Learning, Volume 7*, p. 19, 2012.

[80] J. Deng, Z. Zhang, E. Marchi, and B. Schuller, "Sparse autoencoder-based feature transfer learning for speech emotion recognition," in *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*. IEEE, 2013, pp. 511–516.

[81] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *Association for Computational Linguistics*, 2007.

[82] K. Sarinnapakorn and M. Kubat, "Combining subclassifiers in text categorization: A dst-based solution and a case study," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 12, pp. 1638–1651, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2007.190663

[83] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[84] ——, "Learning to detect unseen object classes by between-class attribute transfer," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[85] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong, "Learning multi-modal latent attributes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[86] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. J. Guibas, and L. Fei-Fei, "Action recognition by learning bases of action attributes and parts," in *IEEE International Conference on Computer Vision*, 2011.

[87] Y. Fu, T. M. Hospedales, T. Xiang, Z. Fu, and S. Gong, "Transductive multi-view embedding for zero-shot recognition and annotation," in *European Conference on Computer Vision*, 2014.

[88] Y. Fu, Y. Yang, T. Hospedales, T. Xiang, and S. Gong, "Transductive multi-label zero-shot learning," in *British Machine Vision Conference*, 2014.

[89] M. Rohrbach, M. Stark, and B. Schiele, "Evaluating knowledge transfer and zero-shot learning in a large-scale setting," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[90] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele, "What helps where – and why? semantic relatedness for knowledge transfer," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 910–917.

[91] R. Socher, M. Ganjoo, H. Sridhar, O. Bastani, C. D. Manning, and A. Y. Ng, "Zero-shot learning through cross-modal transfer," in *Neural Information Processing Systems*, 2013.

[92] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Data and Knowledge Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[93] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, no. 1-2, pp. 151–175, May 2010. [Online]. Available: http://dx.doi.org/10.1007/s10994-009-5152-4

[94] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 1113–1120. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553516

[95] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[96] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *International Conference on Machine Learning*, 2011.

[97] E. A. G. Haibo He, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, 2009.

[98] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *In Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI*, 2000, pp. 111–117.

[99] C. Ling, , C. X. Ling, and C. Li, "Data mining for direct marketing: Problems and solutions," in *In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98*. AAAI Press, 1998, pp. 73–79.

[100] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *Advances in intelligent computing*. Springer, 2005, pp. 878–887.

[101] J. Pengfei, Z. Chunkai, and H. Zhenyu, "A new sampling approach for classification of imbalanced data sets with high density," in *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on*, 2014, pp. 217–222. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6741439

[102] J. Wang, Y. Yao, H. Zhou, M. Leng, and X. Chen, "A new over-sampling technique based on SVM for imbalanced diseases data," in *Mechatronic Sciences, Electric Engineering and Computer (MEC), Proceedings 2013 International Conference on*, 2013, pp. 1224–1228. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6885254

[103] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD 2003*. Springer, 2003, pp. 107–119.

[104] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: the databoost-im approach," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 30–39, 2004.

[105] Z. B. Zabinsky, "Random search algorithms," *Wiley Encyclopedia of Operations Research and Management Science*, 2009.