FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

A STUDY OF SHAPE MODELING AGAINST NOISE

By

CHENG LONG

A Dissertation submitted to the
Department of Statistics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2024

Cheng Long defended this dissertation on November, 8, 2024.
The members of the supervisory committee were:

Adrian Barbu

Professor Directing Dissertation

Xiuwen Liu

University Representative

Anuj Srivastava

Committee Member

Jinfeng Zhang

Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

The following short list of symbols are used throughout the document.

I	the image intensity
$S_{train}^{clean-h}$	the training set of clean horse images, each with dimensions of 128×128
$S_{test}^{clean-h}$	the test set of clean horse images, each with dimensions of 128×128
S_{train}^{salt-h}	the training set of salt and pepper noise horse images, each with dimensions of 128×128
S_{test}^{salt-h}	the test set of salt and pepper noise horse images, each with dimensions of 128×128
$S_{train}^{circle-h}$	the training set of circle noise horse images, each with dimensions of 128×128
$S_{test}^{circle-h}$	the test set of circle noise horse images, each with dimensions of 128×128
S_{train}^{real-h}	the training set of real image noise horse images, each with dimensions of 128×128
S_{test}^{real-h}	the test set of real image noise horse images, each with dimensions of 128×128
$S_{train}^{occlusion-h}$	the training set of occlusion noise horse images, each with dimensions of 128×128
$S_{test}^{occlusion-h}$	the test set of real occlusion noise horse images, each with dimensions of 128×128
$S_{test}^{detection-h}$	the test set of detection image noise horse images, each with dimensions of 128×128
$S_{train}^{probability-h}$	the training set of thresholded probability noise horse images, each with dimensions of 128×128
$S_{test}^{probability-h}$	the test set of thresholded probability noise horse images, each with dimensions of 128×128
S_{test}^{all-h}	the test set of six noise horse images, each with dimensions of 128×128
$S_{train}^{clean-b}$	the training set of clean bird images, each with dimensions of 128×128
$S_{test}^{clean-b}$	the test set of clean bird images, each with dimensions of 128×128
S_{train}^{salt-b}	the training set of salt and pepper noise bird images, each with dimensions of 128×128
S_{test}^{salt-b}	the test set of salt and pepper noise bird images, each with dimensions of 128×128
$S_{train}^{circle-b}$	the training set of circle noise bird images, each with dimensions of 128×128
$S_{test}^{circle-b}$	the test set of circle noise bird images, each with dimensions of 128×128
S_{train}^{real-b}	the training set of real image noise bird images, each with dimensions of 128×128
S_{test}^{real-b}	the test set of real image noise bird images, each with dimensions of 128×128
$S_{train}^{occlusion-b}$	the training set of occlusion noise bird images, each with dimensions of 128×128
$S_{test}^{occlusion-b}$	the test set of real occlusion noise bird images, each with dimensions of 128×128
$S_{test}^{detection-b}$	the test set of detection image noise bird images, each with dimensions of 128×128
$S_{train}^{probability-b}$	the training set of thresholded probability noise bird images, each with dimensions of 128×128
$S_{test}^{probability-b}$	the test set of thresholded probability noise bird images, each with dimensions of 128×128
S_{test}^{all-b}	the test set of six noise bird images, each with dimensions of 128×128
C_0	the background region
C_1	the foreground region

p	the flip probability in salt and pepper noise
r	the radius in circle noise
$S_{train}^{color-h}$	the training set of original color horse images, each with its original dimensions
$S_{train-original}^{clean-h}$	the training set of clean horse images, each with its original dimensions
$S_{test-original}^{clean-h}$	the test set of clean horse images, each with its original dimensions
$S_{train-original}^{real-h}$	the training set of real image noise horse images, each with its original dimensions
$S_{test-original}^{real-h}$	the test set of real image noise horse images, each with its original dimensions
$S_{train-original}^{occlusion-h}$	the training set of occlusion noise horse images, each with its original dimensions
$S_{test-original}^{occlusion-h}$	the test set of occlusion noise horse images, each with its original dimensions
$S_{train-original}^{probability-h}$	the training set of thresholded probability noise horse images, each with its original dimensions
$S_{test-original}^{probability-h}$	the training set of thresholded probability noise horse images, each with its original dimensions
$S_{test-original}^{detection-h}$	the test set of detection image noise horse images, each with its original dimensions
$S_{train-256}^{clean-h}$	the training set of clean horse images, each with dimensions of 256×256
$S_{train-256}^{real-h}$	the training set of real image noise horse images, each with dimensions of 256×256
$S_{train-256}^{occlusion-h}$	the training set of occlusion noise horse images, each with dimensions of 256×256
$S_{train-256}^{probability-h}$	the training set of thresholded probability noise horse images, each with dimensions of 256×256
$S_{train-384}^{real-h}$	the training set of real image noise horse images, each with dimensions of 384×384
$S_{train-384}^{occlusion-h}$	the training set of occlusion noise horse images, each with dimensions of 384×384
$S_{train-384}^{probability-h}$	the training set of thresholded probability noise horse images, each with dimensions of 384×384
$S_{train}^{color-b}$	the training set of original color bird images, each with its original dimensions
$S_{train-original}^{clean-b}$	the training set of clean bird images, each with its original dimensions
$S_{test-original}^{clean-b}$	the test set of clean bird images, each with its original dimensions
$S_{train-original}^{real-b}$	the training set of real image noise bird images, each with its original dimensions
$S_{test-original}^{real-b}$	the test set of real image noise bird images, each with its original dimensions
$S_{train-original}^{occlusion-b}$	the training set of occlusion noise bird images, each with its original dimensions
$S_{test-original}^{occlusion-b}$	the test set of occlusion noise bird images, each with its original dimensions
$S_{train-original}^{probability-b}$	the training set of thresholded probability noise bird images, each with its original dimensions
$S_{test-original}^{probability-b}$	the training set of thresholded probability noise bird images, each with its original dimensions
$S_{test-original}^{detection-b}$	the test set of detection image noise bird images, each with its original dimensions
$S_{train-256}^{clean-b}$	the training set of clean bird images, each with dimensions of 256×256
$S_{train-256}^{real-b}$	the training set of real image noise bird images, each with dimensions of 256×256
$S_{train-256}^{occlusion-b}$	the training set of occlusion noise bird images, each with dimensions of 256×256
$S_{train-256}^{probability-b}$	the training set of thresholded probability noise bird images, each with dimensions of 256×256
$S_{train-384}^{real-b}$	the training set of real image noise bird images, each with dimensions of 384×384
$S_{train-384}^{occlusion-b}$	the training set of occlusion noise bird images, each with dimensions of 384×384

$S_{train-384}^{probability-b}$	the training set of thresholded probability noise bird images, each with dimensions of 384×384

ABSTRACT

Shape modeling is a critical task in computer vision, where the accurate representation and extraction of shape features play a vital role in various applications, from object recognition to medical imaging. Despite significant advancements in machine learning, the ability of shape modeling methods to handle noise—especially under real-world conditions—remains underexplored. This dissertation addresses this gap by providing a comprehensive evaluation of various shape modeling techniques, focusing on their robustness to different types of noise. We simulate realistic noise in both controlled and uncontrolled environments and employ Intersection over Union (IoU) as the primary metric for performance evaluation.

To address certain limitations of existing methods, we introduce PCA-UNet, a novel approach that combines Principal Component Analysis (PCA) and U-Net architectures to achieve superior shape denoising performance. Through extensive experiments, we demonstrate that PCA-UNet outperforms classical methods in various noise scenarios. The findings from this research not only highlight the strengths and weaknesses of current approaches but also provide valuable insights for future developments in robust shape modeling.

CHAPTER 1

INTRODUCTION

The analysis of shape has attracted attention for decades since shape is the most basic visual feature of an object. Due to special conditions, sometimes we cannot obtain other features such as the color or texture of an object besides the shape. Moreover, other features of an object are more easily affected by environmental factors. For example, when an object is illuminated by colored light, its color cannot be accurately observed. Hence, shape can be considered as a more stable feature of an object, and this is why shape analysis plays an essential role in many computer vision tasks. In this thesis, we focus on restoring shapes from noisy shape images, a task that can also be called 'shape denoising'.

To denoise shapes, we need to study how can one represent the shape of an object. In earlier studies, many different methods have been designed to construct shape features such as moments (Flusser and Suk, 1993), shape context (Belongie et al., 2002), curvature (Jalba et al., 2006), and mathematical morphology (Pitas and Venetsanopoulos, 1992).

The method of studying the shape features depends on how to define the shape space. Kendall (1984) considers shapes as k-tuples of points in $\mathbb{R}^d$. Under this notion of shape space, Cootes et al. (1995) proposed the Active Shape Model, which performs a PCA (Principal Component Analysis) on the points of the aligned shapes of the training objects. All shapes are represented using the same number of points, and these points correspond to each other between objects.

In this work, we will consider the shape as represented by a binary image. Considering a shape as a binary image allows many machine learning methods to be applied for shape modeling, such as Restricted Boltzmann Machines (RBM) (Smolensky, 1986), Deep Boltzmann Machines (DBM) (Salakhutdinov and Hinton, 2009) and Centered Convolutional Deep Boltzmann Machines (CDBM) (Yang et al., 2021). Moreover, all deep learning based methods for semantic segmentation can also be used for shape modeling, e.g., U-Net (Ronneberger et al., 2015), DeepLabv3+ (Chen et al., 2018), etc.

1.1 Related Work

The focus of our study is to evaluate and compare the performance of different modeling methods for handling shapes represented by binary images, corrupted by different types of noise. There is a paucity of literature focusing on this topic. However, there is some recent literature containing works about shape modeling.

A Deep Boltzmann Machine (DBM) (Salakhutdinov and Hinton, 2009) is a generative probabilistic model composed of multiple layers of hidden units. DBMs are a type of energy-based model that can capture complex data distributions by learning hierarchical representations. Each layer of the DBM captures increasingly abstract features of the input data, making it well-suited for modeling complex structures such as shapes.

(Yang et al., 2021) proposed a 2D shape modeling method, Centered Convolutional Deep Boltzmann Machines (CDBM), that introduces convolution into a DBM-based shape model. The goal of their work is to generate realistic shapes that are different from all training shapes. In their study, the resulting shapes are generated from test images without any noise. The quality of the generated images is judged subjectively. This evaluation process cannot describe the capability of shape modeling directly. Suppose the resulting image is similar to the initialization of the visible layer, the test image. In that case, it only proves that the model will generate results similar to the initial images, rather than the ability to extract features and model the shape. If the resulting image is quite different from the initialization, it is difficult to say whether it has a good shape belonging to the object being modeled. In our work, we try to recover the object shape from different kinds of noise, which means models have to be able to represent the object shape somehow, and our evaluation criterion is more direct: the closer to the original (unseen) shape, the better. In their work, all models used for comparison are based on Restricted Boltzmann Machines. Our work expands the scope of comparison by adding more methods that include both classical ones such as the Active Shape Model (ASM) and recent ones such as the Energy-based Model (EBM) (Pang et al., 2020).

In the field of recovering shapes from noisy images, the Active Shape Model (ASM) (Cootes et al., 1995) learns a Point Distribution Model (PDM) from training sets of correctly labeled images with point correspondences, then exploits the linear formulation of the PDM in an iterative search procedure to recover the original shape from a noisy test image. In their study, the authors are

more focused on how the parameter of each shape feature influences the final shape. In our work, we pay more attention to the quality of generated shape and use the IoU between the resulting shape and corresponding ground truth shape for evaluation. More importantly, as we will see in our experiments, the method only works in some cases and with a good initialization. Our works construct many types of noise for a more challenging and realistic evaluation.

Another approach related to improving shape recovery from noisy segmentations is Post-DAE (Larrazabal et al., 2020), a post-processing method based on denoising autoencoders (DAEs). Post-DAE is designed to enhance the anatomical plausibility of segmentation masks produced by various segmentation algorithms, such as convolutional neural networks. Once a segmentation mask is obtained, Post-DAE projects it onto this learned manifold, correcting any anatomical inconsistencies. In their work, the authors are more focused on improving current segmentation masks so the noise types are limited. In our work, we introduce many types of noise and evaluate shape denoising performance directly from these noise images.

In addition to external noise or occlusion, the difference in viewing angles will also increase segmentation difficulty. Bryner and Srivastava (2014) proposed an elastic, affine-invariant shape model to segment images of targets subject to perspective skew. Since our work focus on external noise, we use images viewed from the same angles for training and testing.

Since recovering shapes from noisy shape images can be seen as a degenerate version of semantic segmentation, the literature in semantic segmentation is related to our work. Long et al. (2015) proposed the Fully Convolutional Network (FCN), which converts the classification network (Krizhevsky et al., 2012) into a fully convolutional network by replacing all the fully connected layers with convolutions. FCN has two advantages: processing input images of any size without losing spatial information, and reducing computation costs. The U-Net (Ronneberger et al., 2015) adds a decoder that is symmetrical to an encoder based on the FCN architecture, and concatenates feature maps of the same level from the encoder and decoder before performing convolutions. Chen et al. (2014) proposed DeepLab, which adopts atrous convolution and fully connected conditional random fields (CRF). In the improved version, DeepLabv2 (Chen et al., 2017a), an atrous spatial pyramid pooling (ASPP) module was proposed, which consists of multiple parallel atrous convolutional layers with different sampling rates. DeepLabv3 (Chen et al., 2017b) improves the ASPP module and adopts batch normalization and ResNet (He et al., 2016) blocks in training, and the

CRF is deprecated. DeepLabv3+ (Chen et al., 2018), the extended version of DeepLabv3, proposed a new decoder module that takes advantage of feature maps from different levels. The Xception model Chollet (2017) and depth-wise separable convolution are also adopted in it.

(He et al., 2021) introduced the Masked Auto-Encoder (MAE), a self-supervised learning method that masks random patches of input images during pretraining and reconstructs the missing content using a transformer-based decoder. This approach has been shown to be highly effective in various vision tasks, including image restoration and segmentation, by leveraging the underlying structure of the data. MAE offers an advantage in recovering fine details from noisy inputs, making it relevant for the shape denoising problem explored in this work.

Deep Implicit Statistical Shape Models (DISSMs) Raju et al. (2022) represent a novel approach to anatomical structure delineation in medical imaging that combines the strengths of convolutional neural networks (CNNs) with the robustness of traditional statistical shape models (SSMs). DISSMs leverage a deep implicit surface representation, allowing for the creation of a compact, descriptive latent shape space that captures anatomical variance, which is also relevant for the shape denoising problem.

1.2 Our Contributions

This work brings the following contributions:

1. We introduce shape denoising, a fundamental problem in shape modeling that lies at the core of many computer vision and medical imaging applications and has not received enough attention in the literature. Specifically, we distinguish between two types of shape denoising: in controlled environments, where the shapes are aligned in size, and in uncontrolled (wild) environments, where the shapes may vary in size and position.

2. We introduce challenging and realistic noises such as the real image noise, the occlusion noise, the circle noise, the detection noise, and the thresholded probability noise. We quantify the level of difficulty of each noisy shape image using its IoU (Intersection over Union) compared to the original shape.

3. We propose a method to evaluate the performance of different shape modeling methods of recovering the correct shape when corrupted with different types of noises.

4. We adapt multiple existing shape modeling methods to the shape denoising problem and systematically evaluate their performance on recovering the correct shape when corrupted by different types of noise.

5. We introduce a method to incorporate a local PCA shape model as a decoder with a CNN such as a ResNetHe et al. (2016) as encoder, which is used to predict the PCA coefficients for a grid of 32×32 patches. This way the ResNet + PCA model form a UNet-like architecture that has a bottleneck (the PCA coefficients) but it has no skip connections and is more interpretable.

6. We perform extensive experiments to evaluate nine methods including ASM, DBM, CDBM, EBM, Deeplab3+, UNet, DISSM, MAE and PCA-UNet against these types of noise. The findings from our study are that some types of noises are easy to handle while other noises are not. Moreover, some methods can deal successfully with specific types of noise while other methods cannot. We also study the influence of the choice of the tuning parameters for each model, and what is the key reason for the method to perform well. Our study lays down guidelines for future researchers who are interested in this field.

CHAPTER 2

EXISTING METHODS FOR SHAPE MODELING

In this chapter we introduce the existing shape modeling methods that will be used in our experiments.

2.1 Active Shape Model

The Active Shape Model (Cootes et al., 1995) uses labeled points in training images to build a statistical model of a specific shape. When there is a new image with the same shape, the statistical model can be matched to the new image to get the deformed shape for this new image.

2.1.1 Labeling the Training Set

For each image, we label a set of points in an image to represent the shape. Each labeled point represents a specific location on the shape boundary.

Figure 2.1: An example of labeled points in the Weizmann horse dataset.

In Fig. 2.1, we label the fourteen control points with particular application-dependent significance first, then interpolate other points between two control points at equal distances.

2.1.2 Aligning the Training Set

To compare equivalent points from different shapes, all the shapes must to aligned. We denote $\mathbf{x}_i$ as the vector of the N points of the i-th shape in the training set.

$$\mathbf{x}_i = (x_{i0}, x_{i1}, ..., x_{i(N-1)}, y_{i0}, y_{i1}, ..., y_{i(N-1)})^T \tag{2.1}$$

Then we use Procrustes analysis to align all the $\mathbf{x}_i$ of the training set. We denote the transformation result of $\mathbf{x}_i$ as $\mathbf{x}_i^{'}$.

2.1.3 Modeling the Statistics of Aligned Shapes

Combine all the $\mathbf{x}_i^{'}$ into a matrix $\mathbf{X}^{'}$. Compute the column mean of $\mathbf{X}^{'}$ to get the mean shape $\boldsymbol{\mu} = (\boldsymbol{\mu}_x, \boldsymbol{\mu}_y) \in \mathbb{R}^N \times \mathbb{R}^N$. Then using the PCA method on $\boldsymbol{X}^{'} - \boldsymbol{\mu}$ to get all the eigenvectors. Select p largest eigenvectors then have $\mathbf{P} = (\mathbf{P}_x, \mathbf{P}_y), \mathbf{P}_x, \mathbf{P}_y \in \mathcal{M}_{N,p}$, $\mathcal{M}_{N,p}$ being the space of $N \times p$ real matrices.

2.1.4 Fitting the PCA Model

We denote the transformation process from $\mathbf{x}_i^{'}$ to $\mathbf{x}_i$ as $A = (dx, dy, s, \theta)$, with translation (dx, dy) and scale s, rotation θ.

$$A(x, y) = (sx\cos\theta + sy\sin\theta + dx, -sx\sin\theta + sy\cos\theta + dy) \tag{2.2}$$

The PCA shape is controlled by variables $(A, \boldsymbol{\beta})$, where $\boldsymbol{\beta} \in \mathbb{R}^p$ are the PCA coefficients. The PCA shape is

$$S(A, \boldsymbol{\beta}) = A(\boldsymbol{\mu}_x + \mathbf{P}_x\boldsymbol{\beta}, \boldsymbol{\mu}_y + \mathbf{P}_y\boldsymbol{\beta}) = (S_1, ..., S_N)^T \tag{2.3}$$

The actual object parsing is obtained from the PCA shape using a vector of displacements $\mathbf{b} = (d_1, ..., d_N) \in [-d_{max}, d_{max}]^N$ along the normals to the PCA shape. We denote the actual object parsing shape as C.

Using the weighted least squares method (Rogers and Graham, 2002) to update A and β, after N_{it} iterations, we can get the final PCA shape.

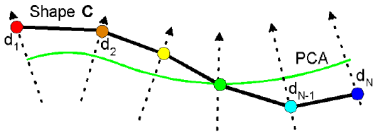


Figure 2.2: PCA shape (green) and the corresponding shape C (black) (Barbu, 2012).

2.2 Boltzmann Machine

2.2.1 Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) (Smolensky, 1986) is a two layer undirected graphical model. It consists of two layers, a visible layer of observed random variables $\mathbf{v} = (v_1, v_2, ..., v_m)$ and one hidden layer of binary hidden random variables $\mathbf{h} = (h_1, h_2, ..., h_n)$, with a full set of connections between them. It is 'restricted' because there is no connection within each layer.

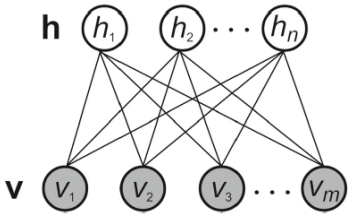


Figure 2.3: Restricted Boltzmann machine (Yang et al., 2021).

The RBM model can also be deemed as a probabilistic model that defines a joint probability distribution over visible and hidden variables. The joint distribution is given by a Gibbs distribution:

$$p(\mathbf{v}, \mathbf{h}; \theta) = \frac{1}{Z(\theta)} e^{-E(\mathbf{v}, \mathbf{h}, \theta)} \tag{2.4}$$

where the energy function is:

$$E(\mathbf{v}, \mathbf{h}, \theta) = -\mathbf{v}^T \mathbf{W} \mathbf{h} - \mathbf{a}^T \mathbf{v} - \mathbf{b}^T \mathbf{h}, \tag{2.5}$$

and $\theta = \{\mathbf{W}, \mathbf{a}, \mathbf{b}\}$ are the model parameters. The weight matrix $\mathbf{W}$ determines the symmetric interaction between pairs of visible and hidden variables, parameters $\mathbf{a}$ and $\mathbf{b}$ are bias vectors for visible and hidden layers respectively, and $Z(\theta) = \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h}; \theta)}$ is a partition function.

The inference between visible and hidden variables in a RBM is:

$$p(\mathbf{h}_j = 1|\mathbf{v}) = \sigma\left(\mathbf{b}_j + \sum_i \mathbf{v}_i \mathbf{W}_{ij}\right), \tag{2.6}$$

$$p(\mathbf{v}_i = 1|\mathbf{h}) = \sigma\left(\mathbf{a}_i + \sum_j \mathbf{W}_{ij} \mathbf{h}_j\right), \tag{2.7}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

2.2.2 Deep Boltzmann Machine

The Deep Boltzmann Machine is obtained by stacking multiple RBMs. Fig. 2.4 displays a two-hidden-layer DBM model, which can be considered as a stack of two RBMs.

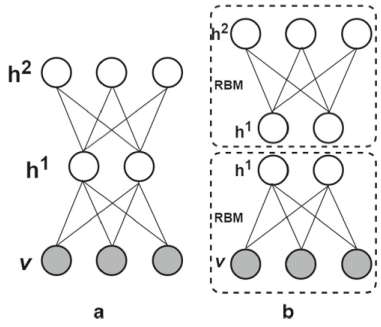


Figure 2.4: Deep Boltzmann machine Yang et al. (2021)

The energy function of this two-hidden-layer DBM is:

$$E(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \theta) = -\mathbf{v}^T \mathbf{W}^1 \mathbf{h}^1 - \mathbf{h}^{1T} \mathbf{W}^2 \mathbf{h}^2 - \mathbf{a}^T \mathbf{v} - \mathbf{b}^T \mathbf{h}^1 - \boldsymbol{c}^T \mathbf{h}^2 \tag{2.8}$$

where $\theta = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{a}, \mathbf{b}, \boldsymbol{c}\}$ are the model parameters. $\mathbf{W}^1$ is the weight matrix between $\mathbf{v}$ and $\mathbf{h}^1$, $\mathbf{W}^2$ is the weight matrix between $\mathbf{h}^1$ and $\mathbf{h}^2$, parameters $\mathbf{a}$, $\mathbf{b}$ and $\boldsymbol{c}$ are bias vectors for $\mathbf{v}$, $\mathbf{h}^1$ and $\mathbf{h}^2$ respectively.

Inference between visible and hidden variables in a two-hidden-layer DBM is:

$$p(\mathbf{v}_i = 1|\mathbf{h}^1) = \sigma\left(\mathbf{a}_i + \sum_j \mathbf{W}^1_{ij} \mathbf{h}^1_j\right), \tag{2.9}$$

$$p(\mathbf{h}_j^1 = 1|\mathbf{v}, \mathbf{h}^2) = \sigma\left(\mathbf{b}_j + \sum_i \mathbf{v}_i \mathbf{W}_{ij}^1 + \sum_m \mathbf{W}_{jm}^2 \mathbf{h}_m^2\right), \tag{2.10}$$

$$p(\mathbf{h}_m^2 = 1|\mathbf{h}^1) = \sigma\left(\boldsymbol{c}_m + \sum_j \mathbf{h}_j^1 \mathbf{W}_{jm}^2\right). \tag{2.11}$$

Instead of treating the visible layer as a vector, if we arrange the visible layer in the form of a 2D matrix, and we connect visible units and hidden units in the first hidden layer with convolution, we get the Convolutional Deep Boltzmann machine(CDBM).

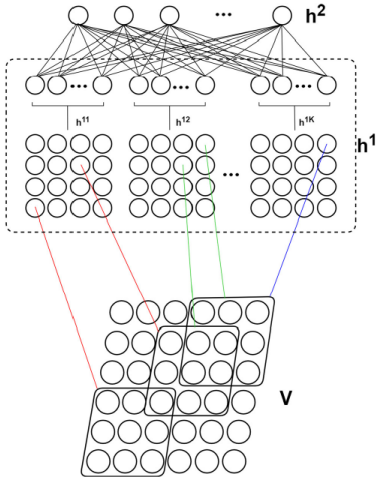


Figure 2.5: Convolutional deep Boltzmann machine (Yang et al., 2021).

2.3 Energy-Based Prior Model

In (Pang et al., 2020), the authors propose a latent space energy-based prior model. Besides of a normal generator model, this model tries to learn an energy-based model for the latent space.

We denote $\mathbf{x}$ as an image, $\mathbf{z} \in \mathbb{R}^d$ as the latent variables. The joint distribution of $(\mathbf{x}, \mathbf{z})$ is

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\alpha}}(\mathbf{z}) p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z}), \tag{2.12}$$

where $p_{\boldsymbol{\alpha}}(\mathbf{z})$ is the prior model with parameters $\boldsymbol{\alpha}$, $p_{\boldsymbol{\beta}}(\mathbf{x}|\mathbf{z})$ is the top-down generation model with parameters $\boldsymbol{\beta}$. Denote $\theta = (\boldsymbol{\alpha}, \boldsymbol{\beta})$.

The prior model $p_{\boldsymbol{\alpha}}(\mathbf{z})$ is formulated as an energy-based model

$$p_{\boldsymbol{\alpha}}(\mathbf{z}) = \frac{1}{Z(\boldsymbol{\alpha})} \exp(f_{\boldsymbol{\alpha}}(\mathbf{z}))p_0(\mathbf{z}), \tag{2.13}$$

where $p_0(\mathbf{z})$ is an isotropic Gaussian distribution, $f_{\boldsymbol{\alpha}}(\mathbf{z})$ is the negative energy and $Z(\boldsymbol{\alpha})$ is the normalization constant.

Short-run MCMC can be used to perform the sampling of $\mathbf{z}$:

$$\mathbf{z}_0 \sim p_0(\mathbf{z}),\ \mathbf{z}_{k+1} = \mathbf{z}_k + s\nabla_{\mathbf{z}} \log \pi(\mathbf{z}_k) + \sqrt{2s}\boldsymbol{\epsilon}_k,\ k = 1, ..., K. \tag{2.14}$$

The learning and sampling algorithm (Pang et al., 2020) is described in Algorithm 1.

Algorithm 1 Learning latent space EBM prior via short-run MCMC

Input: Learning iterations T, learning rate for prior model η_0, learning rate for generation model η_1, initial parameters $\theta_0 = (\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0)$, observed examples $\{\mathbf{x}_i\}_{i=1}^n$, batch size m, number of prior and posterior sampling steps $\{K_0, K_1\}$, and prior and posterior sampling step sizes $\{s_0, s_1\}$.

Output: $\theta_T = (\boldsymbol{\alpha}_T, \boldsymbol{\beta}_T)$.

1: **for** $t = 0 : T - 1$ **do**
2: **Mini-batch**: Sample observed examples $\{\mathbf{x}_i\}_{i=1}^m$.
3: **Prior sampling**: For each $\mathbf{x}_i$, sample $\mathbf{z}_i^- \sim \tilde{p}_{\boldsymbol{\alpha}_t}(\mathbf{z})$ using equation (2.14), where the target distribution $\pi(\mathbf{z}) = p_{\boldsymbol{\alpha}_t}$, and $s = s_0$, $K = K_0$.
4: **Posterior sampling**: For each $\mathbf{x}_i$, sample $\mathbf{z}_i^+ \sim \tilde{p}_{\theta_t}(\mathbf{z}|\mathbf{x}_i)$ using equation (2.14), where the target distribution $\pi(\mathbf{z}) = p_{\theta_t}(\mathbf{z}|\mathbf{x}_i)$ and $s = s_1$, $K = K_1$.
5: **Learning prior model**: $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta_0 \frac{1}{m} \sum_{i=1}^m [\nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}_t}(\mathbf{z}_i^+ - \nabla_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}_t}(\mathbf{z}_i^-]$.
6: **Learning generation model**: $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \eta_1 \frac{1}{m} \sum_{i=1}^m \nabla_{\boldsymbol{\beta}} \log p_{\boldsymbol{\beta}_t}(\mathbf{x}_i|\mathbf{z}_i^+)$.
7: **end for**

2.4 U-Net

The U-Net (Ronneberger et al., 2015) is an improved version of the Fully Convolutional Network (FCN) (Long et al., 2015), and has a better decoding capacity. Fig. 2.6 illustrates the original architecture of the U-Net. It is similar to an encoder-decoder network (Hinton and Salakhutdinov, 2006) with skip connections. In Fig. 2.6, the left side is the Encoder, the right side is the Decoder. In the Encoder, there are four 2×2 max pooling operations with stride 2 for downsampling. Before each downsampling process, two 3×3 convolutions are applied, each convolution being followed by a rectified linear unit (ReLU). In the Decoder, four 2×2 convolutions (up-convolution) are applied

for upsampling. After each upsampling process, the upsampled feature map will be concatenated with the corresponding feature map from the Encoder, and two 3×3 convolutions, each followed by a ReLU, are applied on the new concatenating feature map. In the final layer, a 1×1 convolution is used to map each feature vector to the desired number of classes.

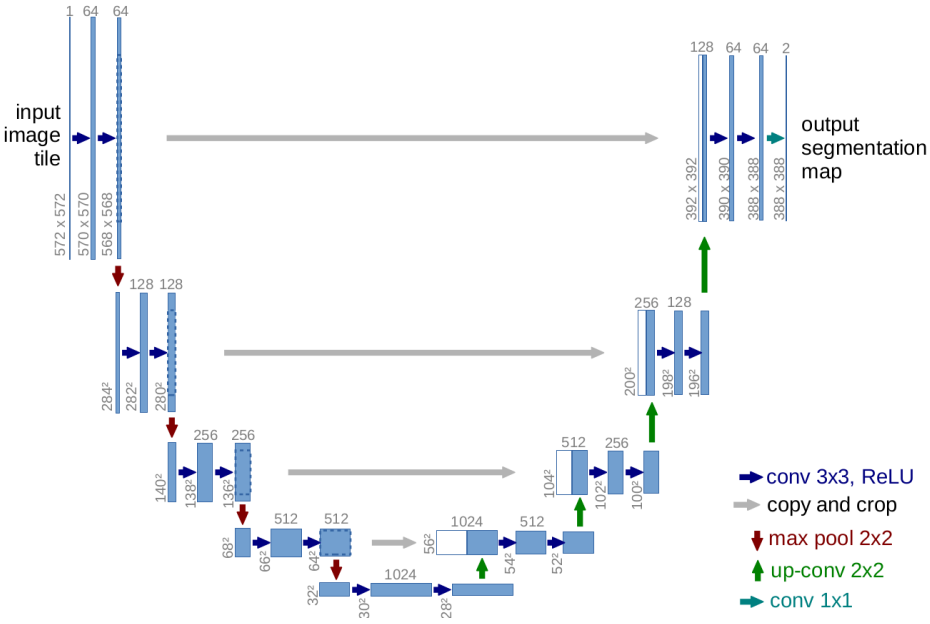
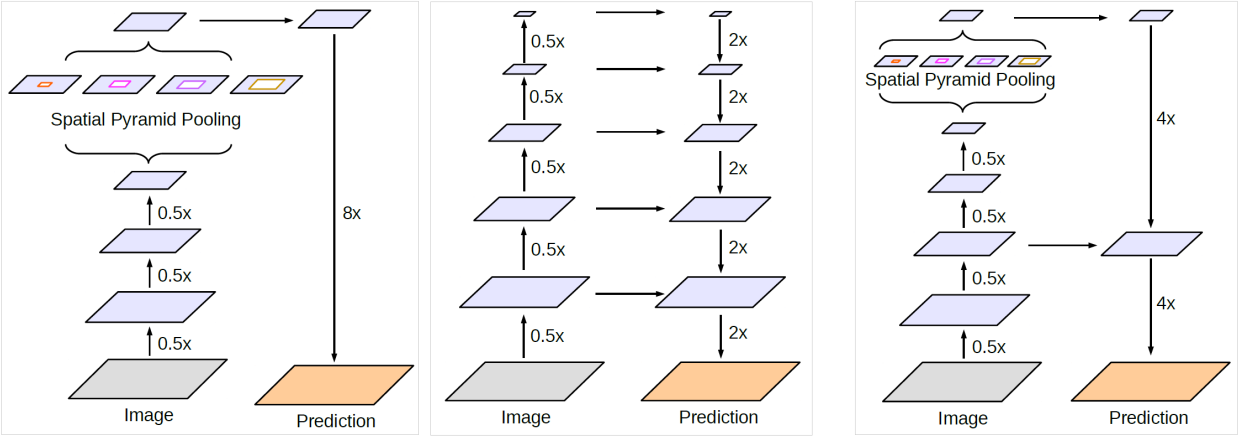


Figure 2.6: Original U-Net architecture (Ronneberger et al., 2015).

2.5 DeepLabv3+

DeepLabv3+ (Chen et al., 2018) is one of the state-of-the-art models for semantic segmentation. It also has the encoder-decoder structure, which gradually captures higher semantic information in the Encoder and recovers spatial information in the Decoder. Spatial pyramid pooling (Grauman and Darrell, 2005; Lazebnik et al., 2006) is applied in DeepLabv3+ by using several parallel atrous convolution with different rates (called Atrous Spatial Pyramid Pooling, or ASPP). Fig. 2.7 illustrates the ideas of Spatial Pyramid Pooling and encoder-decoder structure and how these two methods are used in DeepLabv3+.



(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv

Figure 2.7: Structure improvement from DeepLabv3 to DeepLabv3+ (Chen et al., 2018)

2.6 Masked Autoencoder (MAE)

The Masked Autoencoder (MAE) He et al. (2021) is a simple autoencoding approach that reconstructs the original image given a partial observation, created by randomly masking patches of the image. MAE employs an encoder that maps the observed patches to a latent representation and a lightweight decoder that reconstructs the full image from the latent representation and mask tokens. The encoder only operates on visible, unmasked patches, while the decoder operates on both visible patches and mask tokens. MAE is pre-trained using the mean squared error (MSE) loss to predict the pixel values for each patch. The pre-trained MAE's encoder generates low dimensional representations from images that can generalize well to various downstream computer vision tasks. Fig. 2.8 illustrates the process of pre-training using MAE.

2.7 DISSMs

The Deep Implicit Statistical Shape Models (DISSMs), as proposed by Raju et al. (2022), integrates the superior representation capabilities of deep neural networks with the advantages of statistical shape models (SSMs) to achieve a proficient fitting of realistic shapes to images.

We denote $\mathbf{x} \in \mathbb{R}^2$ as the coordinates in a shape image. The signed distance function (SDF) is

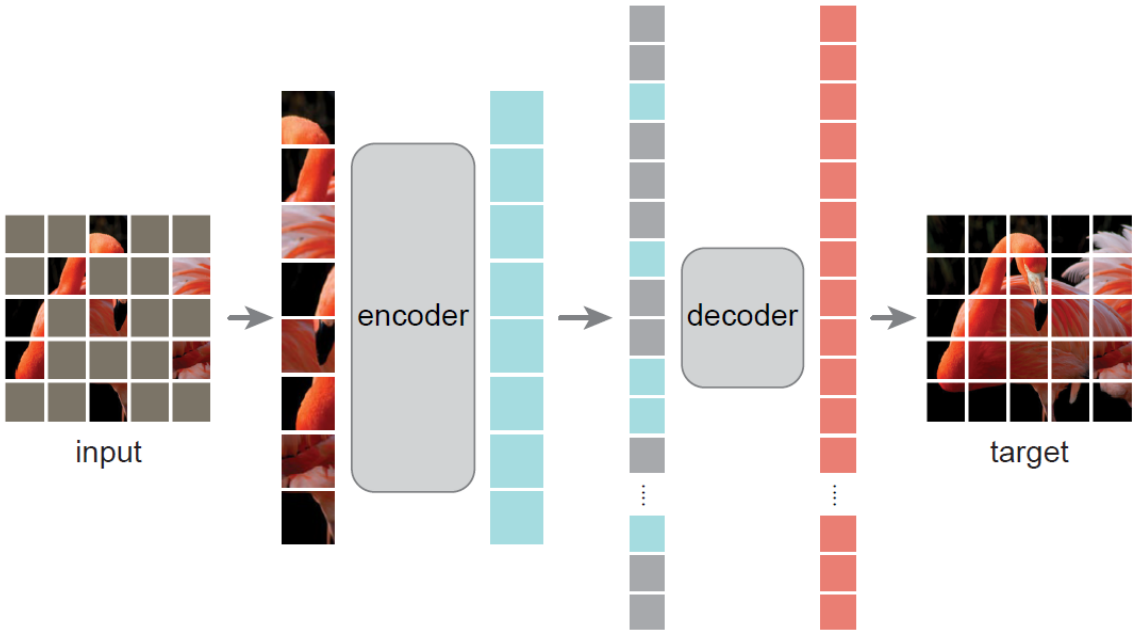


Figure 2.8: Process of pre-training using MAE (He et al., 2021).

$$SDF(\mathbf{x}) = s, \tag{2.15}$$

where $s \in \mathbb{R}$ is the distance to the shape's contour.

The DISSM models a shape by train a deep neural network (decoder) to approximate the shape's SDF.

$$f_{\theta_S}(\tilde{\mathbf{x}}) \approx SDF(\tilde{\mathbf{x}}), \tag{2.16}$$

where θ_S is network weights and $\tilde{\mathbf{x}}$ in a normalized space.

Decoder. The decoder needs two inputs, coordinates $\tilde{\mathbf{x}}$ and a latent vector $\mathbf{z}_k$. The latent vector can specify which shape is being modeled.

To get $\tilde{\mathbf{x}}$, we need to rigidly align each shape. In the rigid alignment, we get each shape's transformation parameters $\boldsymbol{\omega} = \{\mathbf{t}, s\}$, i.e., translation and scale, respectively, where $\mathbf{t} \in \mathbb{R}^2$, $s \in \mathbb{R}$. We denote the rigid transformation as $\mathbf{T}(\boldsymbol{\omega})$.

After alignment, we constructed an affine matrix A to map normalized coordinates to pixel coordinates.

$$\mathbf{x} = A\tilde{\mathbf{x}}, \tag{2.17}$$

then the normalized coordinates are:

$$\tilde{\mathbf{x}} = A^{-1}\mathbf{T}(\boldsymbol{\omega}). \tag{2.18}$$

In the process of training the decoder, the latent vector $\mathbf{z}_k$ is initialized as zeros, then updated by applying auto-decoding latent variables (Park et al., 2019).

Encoder. After the decoder is trained, we need to train an encoder to predict transformation parameters $\boldsymbol{\omega}$ and latent vector $\mathbf{z}_k$. Fig. 2.9 illustrates how Encoder and Decoder combine to produce a result.

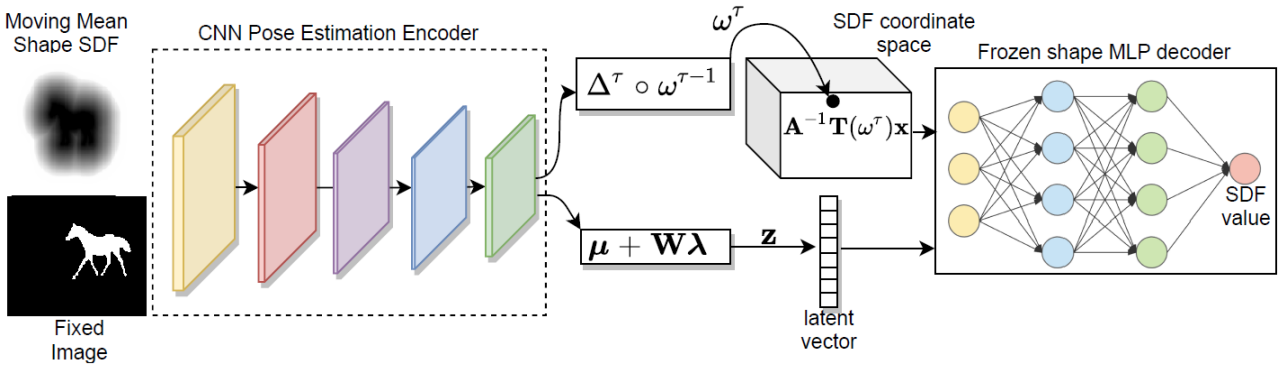
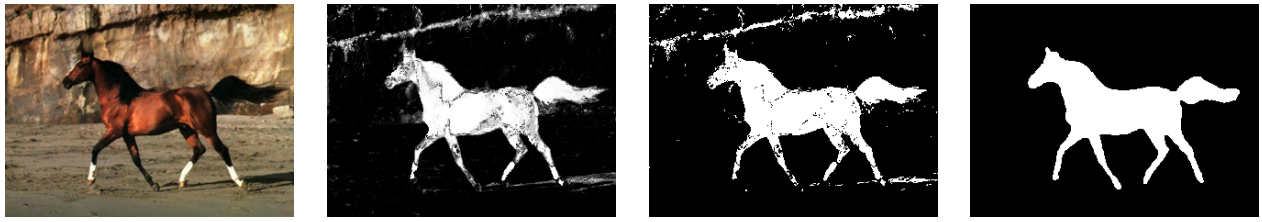


Figure 2.9: Deep implicit statistical shape modeling (DISSM) framework (Raju et al., 2022).

CHAPTER 3

THE SHAPE DENOISING PROBLEM

In this chapter we introduce the shape representation that will be used in our study and six different types of noise that can be used to perturb shapes. We also introduce the shape denoising problem, which involves retrieving a shape from a noisy input shape. For example, a thresholded probability map can be seen as the shape of an an object corrupted by some noise. In this case, shape denoising is to segment the object from the thresholded probability map.



(a) original color image (b) probability map (c) thresholded at 100 (d) denoised shape

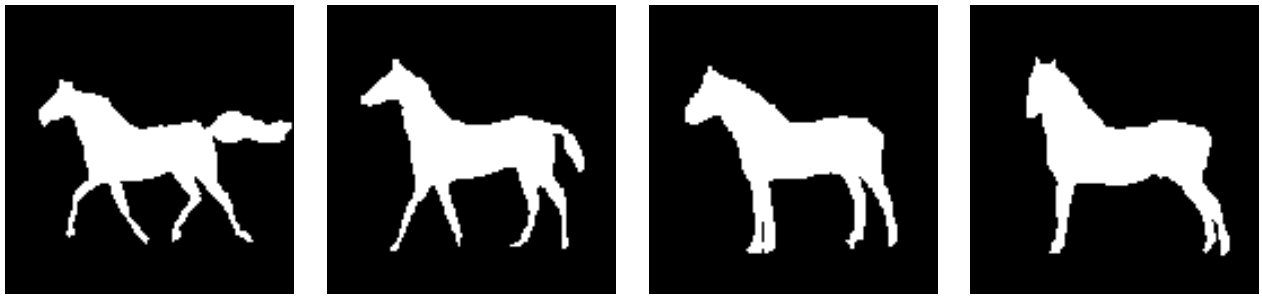
Figure 3.1: Denoising thresholded probability map.

3.1 Shape Representation

In this work shapes will be represented as binary images of a certain size. In our experiments we use 128×128 binary images. All the shape images have black background and white foreground. All foregrounds are in the middle of the images and have approximately similar sizes, obtained as described below. Four examples of shapes used in our study are shown in Figure 3.2.

3.1.1 Shape Alignment

For each binary image I, let $I(x, y) \in \{0, 1\}$ be the intensity at location (x, y). Let C_1 be the region that belongs to foreground, $C_1 = \{(x, y)|I(x, y) = 1\}$ and C_0 the region that belongs to



(a) sample 1 (b) sample 2 (c) sample 3 (d) sample 4

Figure 3.2: Examples of shapes used in this work.

background, $C_0 = \{(x, y)|I(x, y) = 0\}$. The center point of the foreground is

$$(\bar{x}, \bar{y}) = \left(\frac{1}{N} \sum_{i=1}^{N} x_i, \frac{1}{N} \sum_{i=1}^{N} y_i \right) \tag{3.1}$$

where $N = |C_1|$, $(x_i, y_i) \in C_1, i = 1, ..., N$. Let $d(x_i, y_i)$ be the distance between point $(x_i, y_i) \in C_1$ and the center point $(\bar{x}, \bar{y})$,

$$d(x_i, y_i) = \|(x_i - \bar{x}, y_i - \bar{y}).\|_2 \tag{3.2}$$

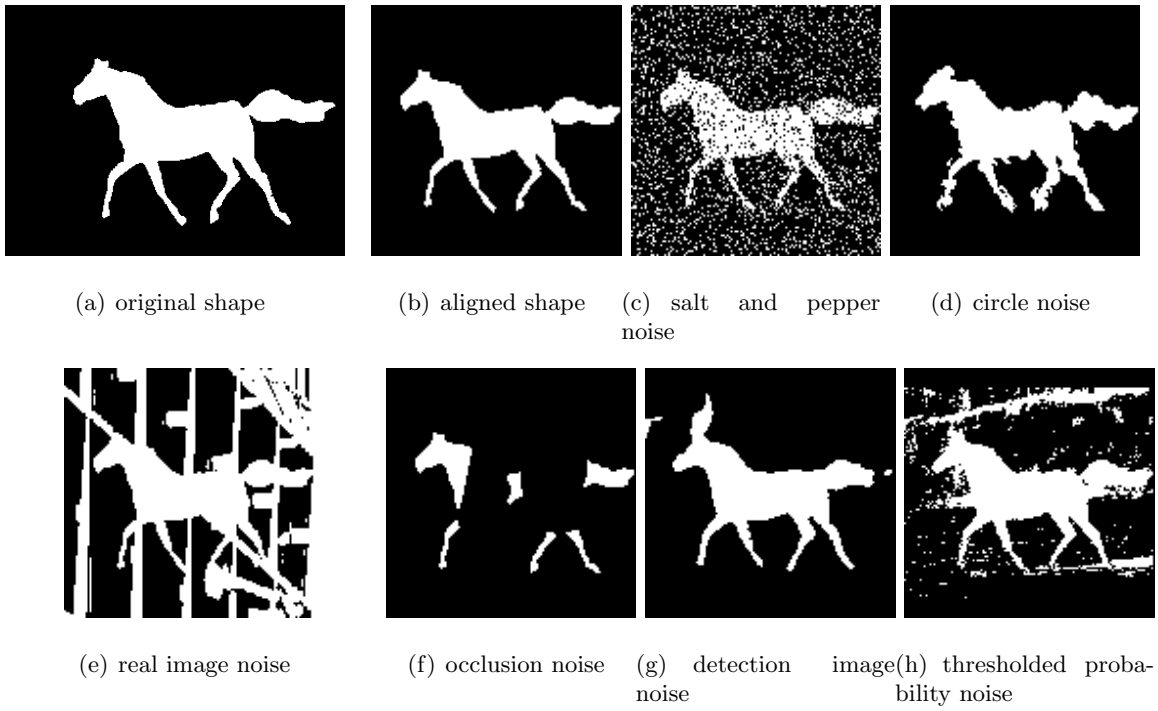
The 80% percentile of the set $D = \{d(x_i, y_i)|i = 1, ..., N\}$ is computed and denoted by p_{80}. The binary image is then rescaled using the scale factor $40/p_{80}$. The image is rescaled using bicubic interpolation, obtaining a grayscale image, which is then thresholded to obtain a binary image again. Denote by $I^{'}$ the rescaled image, $I^{'}(x, y) \in \{0, 1\}$, with $C_1^{'} = \{(x, y)|I^{'}(x, y) = 1\}$ as the foreground region and $C_0^{'} = \{(x, y)|I^{'}(x, y) = 0\}$ as the background region. The new center point of the foreground is

$$(\bar{x}^{'}, \bar{y}^{'}) = \left(\frac{1}{N^{'}} \sum_{i=1}^{N^{'}} x_i, \frac{1}{N^{'}} \sum_{i=1}^{N^{'}} y_i \right) \tag{3.3}$$

where $N^{'} = |C_1^{'}|$, $(x_i, y_i) \in C_1^{'}, i = 1, ..., N^{'}$. The image I' is then cropped to obtain an image of size 128×128 centered at $(\bar{x}^{'}, \bar{y}^{'})$. Padding is used if necessary. Examples of centered and rescaled images obtained with the above procedure are shown in Figure 3.2.

3.2 Introducing Noise in Shapes

We will introduce six different types of noise that could be used for perturbing shapes for the purpose of evaluating a shape denoising method on its capability of recovering the original shape. The six types of noise are salt and pepper noise, circle noise, real image noise, occlusion noise, detection image noise and thresholded probability noise, as illustrated in Fig. 3.3.



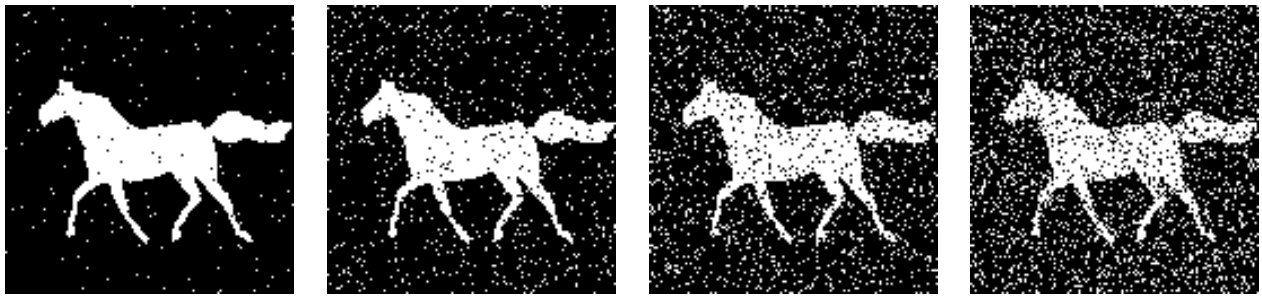
(a) original shape (b) aligned shape (c) salt and pepper noise (d) circle noise

(e) real image noise (f) occlusion noise (g) detection image noise (h) thresholded probability noise

Figure 3.3: Illustration of shape alignment and the six types of shape noise introduced in this work.

3.2.1 Salt and Pepper Noise

The salt and pepper noise is obtained by flipping each pixel to its opposite value with a probability p. For this reason it could also be called Bernoulli noise. The flipping probability p is used to control the noise level. Examples of a shape with different levels of salt and pepper noise are shown in Figure 3.4.



(a) $p = 0.01$ (b) $p = 0.05$ (c) $p = 0.10$ (d) $p = 0.15$

Figure 3.4: Salt and pepper noise with different levels p.

3.2.2 Circle Noise

The circle noise is obtained by adding semicircles or punching holes at random locations on the boundary between the foreground and the background. The radius r of the semicircles or holes is used to control the noise level. For each noise image, the radius is fixed. Noise images with different values of r are illustrated in Figure 3.5.

(a) $r = 1$ (b) $r = 3$ (c) $r = 6$ (d) $r = 10$

Figure 3.5: Circle noise with different levels.

3.2.3 Real Image Noise

Real images are binarized by thresholding with various thresholds and the obtained binary image is used to replace the background pixels of a clean shape. Examples of obtained noisy shapes using this method are shown in Figure 3.6. In this case the noise level cannot be controlled using

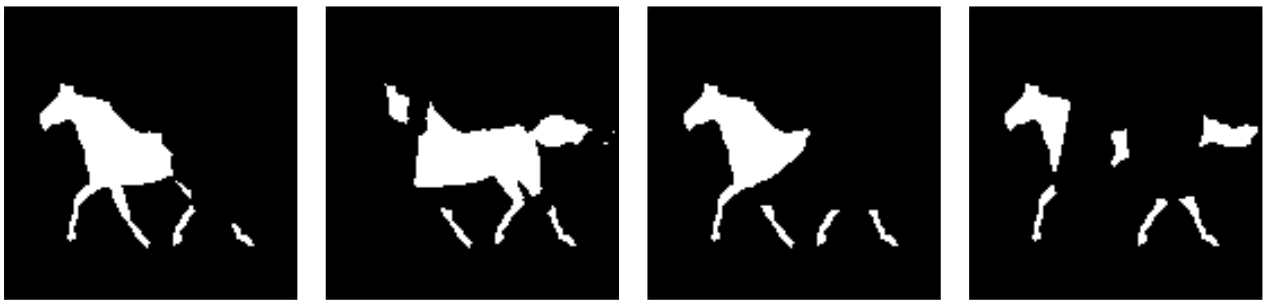
parameters, but the obtained images can be organized by the IoU defined in Equation (3.6) below or by the percent of the object boundary that is left intact.

(a) sample 1 (b) sample 2 (c) sample 3 (d) sample 4

Figure 3.6: Samples of real image noise.

3.2.4 Occlusion Noise

Noise is obtained by randomly occluding a part of the shape. Examples of obtained noisy shapes using this method are shown in Figure 3.7. The obtained images can be organized by the IoU defined in Equation (3.6) below or by the percent of the object boundary that is left intact.



(a) sample 1 (b) sample 2 (c) sample 3 (d) sample 4

Figure 3.7: Samples of occlusion noise.

3.2.5 Detection Image Noise

If the shape has been obtained as the segmentation of an object from a color or grayscale image, a segmentation algorithm such as a Fully Convolutional Network is used to extract the predicted segmentation and this segmentation is the detection image noise.

(a) object shape (b) color image (c) detection image noise

Figure 3.8: Sample of detection image noise in the Weizmann horse dataset.

3.2.6 Thresholded Probability Noise

For any binary mask image M, the corresponding original color image I is used to generate a probability map as follows. Let $I(x, y) \in \{0, 1, ..., 255\}^3$ be the image intensity at location (x, y). Denote N as the number of pixels in the color image, the intensities of all pixels can be represented by a N points in $\mathbb{R}^3$. The k-means clustering method is used to partition the N points into k clusters, obtaining cluster indices for all pixels $L \in \{1, 2, ..., k\}^N$. Denote $C_1 = \{(x, y), M(x, y) = 1\}$ as the foreground region and $C_0 = \{(x, y), M(x, y) = 0\}$ as the background region. For cluster $i \in \{1, 2, ..k\}$, the number of pixels from this cluster that belong to foreground region or background region are computed:

$$\begin{aligned} N_{i1} &= |\{(x, y)|L(x, y) = i \wedge (x, y) \in C_1\}|, \\ N_{i0} &= |\{(x, y)|L(x, y) = i \wedge (x, y) \in C_0\}|. \end{aligned} \tag{3.4}$$

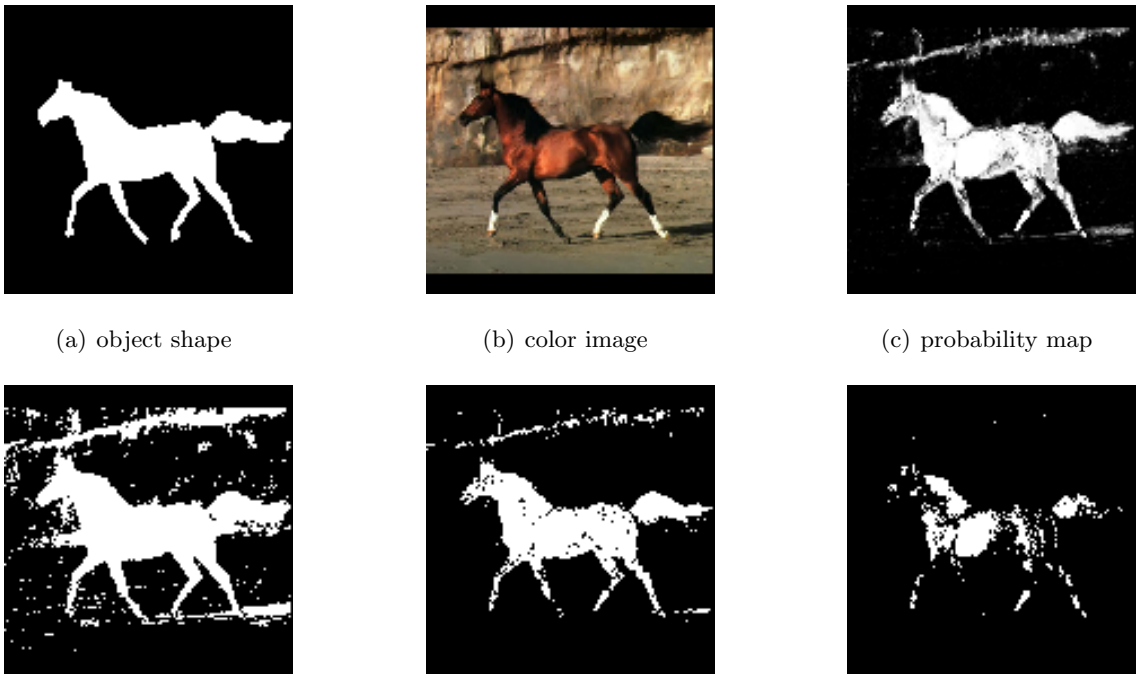
Then the probability map of color image I can be computed as follows:

$$P(x, y) = \frac{N_{j1}}{N_{j1} + N_{j0}}, \text{ where } j = L(x, y). \tag{3.5}$$

Then binary noisy shapes are obtained by applying different thresholds to the probability map. Examples of obtained probability maps with different thresholds are shown in Figure 3.9.

3.3 Evaluation Measures for Shape Denoising

Shape denoising is the process of removing the noise from a shape image, with the goal of obtaining a shape as close to the original shape as possible. Figure 3.10 illustrates a example of shape denoising.



(a) object shape (b) color image (c) probability map

(d) threshold 0.04 (e) threshold 0.5 (f) threshold 0.98

Figure 3.9: Example of a probability map and noisy shapes obtained with different thresholds.

There are different measures to evaluate the denoising performance by using denoising result and mask image.

IoU. Intersection over Union (IoU), also known as the Jaccard Index, is a measure of similarity for two sets, with a range from 0% to 100%. The higher the percentage, the more similar the two sets. Denoting I as the mask image intensity, $I(x, y) \in \{0, 1\}$, let C_1 be the region belonging to the object, $C_1 = \{(x, y)|I(x, y) = 1\}$. Using I_r to denote the denoising result image intensity, $I_r(x, y) \in \{0, 1\}$ let C_r be the region belonging to the foreground, $C_r = \{(x, y)|I_r(x, y) = 1\}$. Then the IoU is:

$$IoU(C_1, C_r) = \frac{|C_1 \cap C_r|}{|C_1 \cup C_r|} = \frac{|C_1 \cap C_r|}{|C_1| + |C_r| - |C_1 \cap C_r|}. \tag{3.6}$$

Dice. Dice coefficient, also known as F1 Score, is a measure similar to IoU, having a range from 0% to 100% too. The calculation of Dice is:

$$Dice(C_1, C_r) = \frac{2 * |C_1 \cap C_r|}{|C_1| + |C_r|}. \tag{3.7}$$

(a) original shape (b) noisy shape (c) denoised shape

Figure 3.10: Shape denoising example. The noisy shape (b) has been obtained from the original shape (a) by a noise inducing process such as those described in Section 3.2. A shape denoising method such as those described in Section 2 is used to obtain the denoised shape (c).

Hausdorff distance. Hausdorff distance, also called Pompeiu–Hausdorff distance, measures how far two subsets of a metric space are from each other. Denoting the boundaries of C_1 and C_0 with δC_1 and δC_0, the Hausdorff distance between δC_1 and δC_0 is:

$$d_H(\delta C_1, \delta C_0) = \max\left\{ \sup_{a\in\delta C_1} \inf_{b\in\delta C_0} \mathrm{d}(a,b), \sup_{b\in\delta C_0} \inf_{a\in\delta C_1} \mathrm{d}(a,b) \right\}. \tag{3.8}$$

CHAPTER 4

PCA-UNET

PCA-UNet is the proposed approach that combines Principal Component Analysis (PCA) and the U-Net architecture to improve shape modeling and denoising. By incorporating PCA, which effectively reduces dimensionality and retains critical shape features, PCA-UNet leverages the strengths of both techniques to achieve enhanced performance in shape denoising tasks.

4.1 Online PCA

PCA can be computed online from an arbitrarily large number of training examples (e.g. obtained by data augmentation), in one pass, without needing to store the data. For that, running averages (RAVE) Kushner and Yin (2003); Sun et al. (2024) can be used to obtain the exact value of the covariance matrix on all the training examples.

The running averages for a set of observations $\mathbf{x}_i \in \mathbb{R}^d, i = 1, ..., n$ are defined as $(n, \mathbf{M}_x, \mathbf{M}_{xx})$ where:

$$\mathbf{M}_x = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \quad \mathbf{M}_{xx} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T. \tag{4.1}$$

These running averages can be updated online from a batch of b observations organized as rows in a matrix $\mathbf{B}$ as:

$$\mathbf{M}_x \leftarrow \frac{n\mathbf{M}_x}{n+b} + \frac{\mathbf{B}^T\mathbf{1}_b}{n+b},\ \mathbf{M}_{xx} \leftarrow \frac{n\mathbf{M}_{xx}}{n+b} + \frac{\mathbf{B}^T\mathbf{B}}{n+b}, \tag{4.2}$$

where $\mathbf{1}_b = (1, ..., 1)^T$ is the vector of ones of size b.

From the running averages, the sample covariance matrix can be computed exactly as:

$$\hat{\mathbf{\Sigma}} = \frac{n}{n-1}(\mathbf{M}_{xx} - \mathbf{M}_x\mathbf{M}_x^T), \tag{4.3}$$

and the sample mean is $\hat{\boldsymbol{\mu}} = \mathbf{M}_x$. From these quantities the PCA can be easily obtained by SVD.

4.2 Modeling Shapes using PCA

Similar to Long and Barbu (2022), in this paper, shapes are represented as binary images of a predefined size, e.g. 256×256 pixels, with foreground value 1 and background value 0. This is a different representation than the Active Shape Models (Cootes et al., 1995) where the shapes are represented as a chain of contour points that have correspondence across shapes.

Shape alignment. The shapes are aligned to be roughly the same size and centered at the same location, similar to Long and Barbu (2022). For a shape (binary image) $\mathbf{S}$, let $\mathbf{S}(x, y) \in \{0, 1\}$ be the pixel intensity value at coordinate (x, y). The shape's center is defined as its center of mass:

$$\bar{\mathbf{S}} = (\bar{\mathbf{x}}, \bar{\mathbf{y}}) = (\frac{1}{|F|} \sum_{(x,y)\in F} x, \frac{1}{|F|} \sum_{(x,y)\in F} y), \tag{4.4}$$

where $F = \{(x, y), \mathbf{S}(x, y) = 1\}$ are the foreground pixels.

The shape's scale is defined as the 80 percentile of the set of distances of the foreground pixels to the center:

$$D = \{\|\mathbf{x} - \bar{\mathbf{S}}\|, \mathbf{x} = (x, y) \in F\}.$$

For 256×256 images, the training shapes are aligned by resizing the image so that the scale is 80 (for the Weizmann Horse dataset (Borenstein et al., 2004)) or 60 (for the Caltech-UCSD Birds 200 dataset (Welinder et al., 2010)), then by translation so that the center is at location $(128, 128)$.

PCA computation. The PCA shape model is obtained as described in Algorithm 2. The algorithm involves a number N^{it} of iterations of perturbing all the shapes using random rotation, translation and scaling, and updating the RAVEs.

Algorithm 2 Shape model training by online PCA

Input: Aligned training shapes $\{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_k\}$

Output: Trained PCA model $\theta = (\boldsymbol{\mu}, \mathbf{P}) \in \mathbb{R}^d \times \mathbb{R}^{d\times q}$

1: Initialize RAVE as $(n, \mathbf{M}_x, \mathbf{M}_{xx}) = (0, \mathbf{0}, \mathbf{0})$
2: **for** $i = 1$ to N^{it} **do**
3: Perturb the training shapes, obtaining vectorized perturbed shapes $\mathbf{x}_1, ..., \mathbf{x}_k \in \mathbb{R}^d$.
4: Update RAVE $(n, \mathbf{M}_x, \mathbf{M}_{xx})$ using Eq. (4.2) with $\mathbf{B} = (\mathbf{x}_1, ..., \mathbf{x}_k)^T$.
5: **end for**
6: Compute $\hat{\boldsymbol{\Sigma}} = \frac{n}{n-1}(\mathbf{M}_{xx} - \mathbf{M}_x\mathbf{M}_x^T)$
7: Decompose $\hat{\boldsymbol{\Sigma}} = \mathbf{V}\mathbf{D}\mathbf{V}^T$ by SVD
8: Obtain $\boldsymbol{\mu} = \mathbf{M}_x$, and $\mathbf{P}$ as the first q columns of $\mathbf{V}$.

The RAVE for a $128 \times 128 = 2^{16}$ pixel image contains the matrix $\mathbf{M}_{xx}$ of size $d \times d$ with $d = 2^{16} = 16,384$, which is quite large and non-sparse.

Grid PCA. For more modeling flexibility and accuracy and to be able to handle large images, the shape space (of binary images) can be divided into a grid of non-overlapping squares (patches) and each square can be modeled by PCA.

In our application, the 256×256 shape images are divided in $8 \times 8 = 64$ squares of size 32×32 each. This way one RAVE (or a grid of $8 \times 8 = 64$ RAVEs) with $d = 1024$ is used in Algorithm 2 instead of one RAVE with $d = 65535$.

PCA Shape Reconstruction. A reconstructed binary PCA shape is obtained as $\hat{\mathbf{y}} = I(\hat{\mathbf{S}} > 0.5)$, where $I(\cdot)$ is the indicator function ($I(v) = 1$ if v is true, otherwise 0) applied elementwise, and

$$\hat{\mathbf{S}} = \boldsymbol{\mu} + \mathbf{P}\mathbf{c}, \tag{4.5}$$

where $\boldsymbol{\mu}$ is the vectorized mean, $\mathbf{P}$ are the vectorized PCs, and $\mathbf{c} \in \mathbb{R}^q$ is a parameter vector.

Similarly, given a grid of parameter vectors $\mathbf{c}_{ij} \in \mathbb{R}^q$, $(i, j) \in \{1, ..., K\} \times \{1, ...L\}$, Eq. (4.5) can be used to reconstruct corresponding patches

$$\hat{\mathbf{S}}_{ij} = \boldsymbol{\mu}_{ij} + \mathbf{P}_{ij}\mathbf{c}_{ij}, (i, j) \in \{1, ..., K\} \times \{1, ...L\}, \tag{4.6}$$

which are placed at the appropriate grid locations (i, j) to obtain the whole reconstructed shape $\hat{\mathbf{y}}$ after thresholding.

Here a grid of PCAs $(\boldsymbol{\mu}_{ij}, \mathbf{P}_{ij})$ can be used for aligned images of the same size (see Figure 4.1), or a single PCA $(\boldsymbol{\mu}, \mathbf{P})$ can be used on the grid of patches

$$\hat{\mathbf{S}}_{ij} = \boldsymbol{\mu} + \mathbf{P}\mathbf{c}_{ij}, (i, j) \in \{1, ..., K\} \times \{1, ...L\}, \tag{4.7}$$

to obtain the whole $\hat{\mathbf{S}}$ and the reconstructed shape $\hat{\mathbf{y}} = I(\hat{\mathbf{S}} > 0.5)$. Experiments from Section 6.5.1 will reveal that a single PCA $(\boldsymbol{\mu}, \mathbf{P})$ is as effective as a grid of PCAs for modeling object shapes using a grid of 32×32 pixel patches.

4.3 Architecture and Training

In PCA-UNet, a fully convolutional neural network (FCNN) such as a ResNet (He et al., 2016) is used as a feature generator for predicting the PCA coefficients for each patch, as illustrated in Figure 4.1.

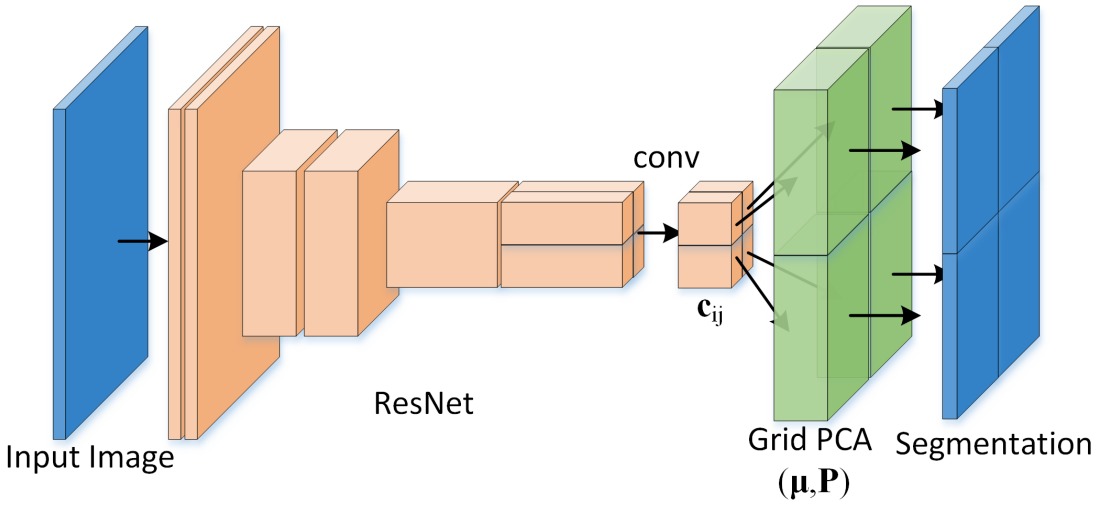


Figure 4.1: Diagram of the PCA-UNet.

The size of the patches needs to align with the output of the FCNN. For example, a ResNet-50 obtains from an input image of size 256×384 a feature vector of size $8 \times 12 \times 2048$ before the average pooling layer. In this case, the shape is decomposed into a grid of 8×12 patches of size 32×32 each. Then the PCA coefficients are predicted from the CNN outputs using a 1×1 convolutional layer. For example, if the 32×32 shape patches are represented with a PCA $(\boldsymbol{\mu}, \mathbf{P})$ with $q = 64$ PCs, then the convolutional layer has size $1 \times 1 \times 64$, obtaining a $8 \times 12 \times 64$ output of predicted PCs $\mathbf{c}_{i,j} \in \mathbb{R}^{64}, (i,j) \in \{1,..,8\} \times \{1,..,12\}$ from the $8 \times 12 \times 2048$ output of the ResNet-50. Optionally, an attention layer (Vaswani et al., 2017) could be introduced before the convolution layer.

Then the PCA model $(\boldsymbol{\mu}, \mathbf{P})$ is used to reconstruct the corresponding patches of the output segmentation from the predicted PCs using Eq. (4.7).

The PCA-UNet's trainable parameters are $(\mathbf{R}, \mathbf{C})$, consisting of the CNN Layers $\mathbf{R}$ and the added convolutional layer $\mathbf{C}$, which together form a FCNN $f(\mathbf{x}; \mathbf{R}, \mathbf{C}) = (\mathbf{c}_{11}(\mathbf{x}; \mathbf{R}, \mathbf{C}), ..., \mathbf{c}_{KL}(\mathbf{x}; \mathbf{R}, \mathbf{C}))$. Observe that the PCA model $(\boldsymbol{\mu}, \mathbf{P})$ is frozen. Training is done in an end-to-end fashion by standard backpropagation using the IOU Loss, which for a training example $(\mathbf{x}, \mathbf{y})$ is:

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\sum_j \hat{\mathbf{y}}_j \mathbf{y}_j}{\sum_j \max(\hat{\mathbf{y}}_j, \mathbf{y}_j)}, \tag{4.8}$$

where the sums are taken over the pixels, $\hat{\mathbf{y}} = \sigma(\alpha(\hat{\mathbf{S}}))$, with $\sigma(\cdot)$ as the sigmoid, α is a tuning parameter ($\alpha = 50$ in our experiments), and $\hat{\mathbf{S}}$ is obtained as in Eq (4.5), or reconstructed from patches as in Eqns. (4.6) and (4.7).

CHAPTER 5

EVALUATION OF SHAPE DENOISING METHODS

In this section we evaluate the shape denoising methods described in Chapters 2 and 4 for denoising shapes corrupted by the six types of noise introduced in Section 3.2. The criterion we use to estimate the quality of the denoising result is the IoU (3.6).

5.1 Dataset Construction

5.1.1 Clean Image Datasets

We use the Weizmann Horse dataset (Borenstein et al., 2004) and the Caltech-UCSD Birds 200 dataset (Welinder et al., 2010) for our experiments. The Weizmann Horse dataset contains 327 horse images and their corresponding mask images. All mask images were aligned as described in section 3.1 to have all shapes centered and of approximately the same size. The aligned images were resized to 128×128. Examples of input mask images and resulting aligned images are shown in Figure 5.1.

(a) mask image1 (b) aligned image1 (c) mask image2 (d) aligned image2

Figure 5.1: Two alignment examples of horse dataset.

From the 327 aligned horse images,159 images were randomly selected as the training set $S_{train}^{clean-h}$ and the other 168 images as the test set $S_{test}^{clean-h}$.

We use 417 images of seven Flycatcher species in the bird dataset, 207 images were randomly selected as the training set $S_{train}^{clean-b}$ and the other 210 images as the test set $S_{test}^{clean-b}$.

5.1.2 Noisy Image Datasets

Training Sets. For each image in training set $S_{train}^{clean-h}$, noisy versions were generated as follows:

- salt and pepper noise was generated using 16 flipping probabilities $p \in \{0, 0.01, 0.02, ..., 0.15\}$. For each probability, ten different samples were generated, for a total of 25440 training salt and pepper noise images S_{train}^{salt-h}.

- circle noise was generated using 11 radii $r \in \{0, 1, 2, ..., 10\}$. For each radius, ten samples were generated for a total of 17490 circle noise training images $S_{train}^{circle-h}$.

- real image noise was generated by randomly selecting a 128×128 size random patch from a randomly selected image of the PASCAL VOC 2012 Dataset (Everingham et al., 2010). The patch was converted to grayscale, and different thresholds in $\{10, 40, 70, 100, 130, 160, 190, 220, 250\}$ were used on the grayscale patch to generate nine binary images. These binary images were used as the background of the selected training image. By this method, parts of the generated binary images are almost all white. Since we can not extract shape features from images with no shape information, we need to quantify how much shape information is left in the generated images. For that purpose we compute the ratio between the length of the remaining correct boundary after adding noise and the length of the original boundary. The generated images with the remaining boundaries ratio above 0.5 are added to the real image noise training set S_{train}^{real-h}, which contains 27009 images.

- occlusion noise was generated by randomly selecting a 128×128 size random patch from a randomly selected mask image of the PASCAL VOC 2012 Dataset. The binary patch was used to occlude the foreground of the selected training image. The generated images with the remaining boundaries ratio above 0.5 are added to the occlusion noise training set $S_{train}^{occlusion-h}$, which contains 4747 images.

- thresholded probability noise sets are generated as described in Section 3.2.6. For each image in the train set $S_{train}^{clean-h}$, we use the corresponding original color image in the Weizmann Horse dataset to build a train set $S_{train}^{color-h}$. The pixels on each color image in $S_{train}^{color-h}$ are clustered into 800 clusters using k-means clustering. Based on the clustering result and the corresponding mask image in $S_{train}^{clean-h}$, we compute the probability that each pixel is in the object and generate a probability map of this image. For each grayscale probability map, we use different thresholds in $\{10/255, 20/255, 30/255, ..., 250/255\}$ to generate binary noise images. All the binary images build the thresholded probability noise set $S_{train}^{probability-h}$.

For bird images in training set $S_{train}^{clean-b}$, use same methods to generate S_{train}^{salt-b} (33120 images), $S_{train}^{circle-b}$ (22770 images), S_{train}^{real-b} (64406 images), $S_{train}^{occlusion-b}$ (5054 images) and $S_{train}^{probability-b}$ (5175 images).

Test Sets. For each image in test set $S_{test}^{clean-h}$, we use the above methods to generate noisy images, obtaining the salt and pepper noise set S_{test}^{salt-h}, circle noise set $S_{test}^{circle-h}$, real image noise S_{test}^{real-h}, occlusion noise $S_{test}^{occlusion-h}$ and thresholded probability noise set $S_{test}^{probability-h}$. We organize the noisy images by their IoU with the original image. For each IoU level (e.g. 0.5-0.6, 0.6-0.7, ...) we randomly select at most 1000 images for testing.

To generate detection noise, the color images in $S_{train}^{color-h}$ are resized to 50% of their original size. Then all patches of size 33×33 satisfying one of the following conditions are generated. 1.The center point of the patch is inside the object from the mask image. 2. The center point of the patch is outside the object, but the distance between the center point and object boundary is in the range of 2 and 50. This way we obtain 1,262,362 patches, containing 353,334 positive (inside) patches and 909,028 negative(outside) patches. A Fully Convolutional Network (FCN) is designed to output one number when the input size is 33×33. We use the 1,262,362 patches as a training set to train the FCN and obtain a model M_1. Using M_1 as initialization, using the full 50% size images instead of 33×33 patches as inputs, the FCN is fine-tuned obtaining model M_2. Probability maps are generated using the predicted outputs from M_2. The probability map is added as the fourth channel besides the R, G, B channels of a color image. The same method to generate 25×25 patches from the four-channel data, obtaining 1,256,087 patches, containing 317,895 positive(inside) patches and 938,192 negative(outside) patches. A new FCN is designed to output one number when the input size is 25×25. Then the 1,256,087 patches are used to train the new FCN and obtain a new model M_3. Using M_3 as initialization and the four-channel images as inputs, the new FCN is fine-tuned obtaining the final model M_4. For each image in the test set $S_{test}^{clean-h}$, the pre-trained models M_2 and M_4 are applied to generate the detection image noise set $S_{test}^{detection-h}$.

We denote the set $\{S_{test}^{salt-h}, S_{test}^{circle-h}, S_{test}^{real-h}, S_{test}^{occlusion-h}, S_{test}^{detection-h}, S_{test}^{probability-h}\}$ as S_{test}^{all-h}. Generate test sets $\{S_{test}^{salt-b}, S_{test}^{circle-b}, S_{test}^{real-b}, S_{test}^{occlusion-b}, S_{test}^{detection-b}, S_{test}^{probability-b}\}$ on the bird dataset, denote these six sets as S_{test}^{all-b} Table 5.1 shows the number of images in each category on S_{test}^{all-h} and S_{test}^{all-b}.

Table 5.1: Number of noisy test images by noise category.

Test set \ IoU(begin)	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
S_{test}^{salt-h}	1000	1000	1000	1000	1000
$S_{test}^{circle-h}$	77	1000	1000	1000	1000
S_{test}^{real-h}	1000	1000	1000	1000	1000
$S_{test}^{occlusion-h}$	479	622	690	938	1779
$S_{test}^{detection-h}$	3	7	14	77	60
$S_{test}^{probability-h}$	298	673	1095	1345	525
S_{test}^{salt-b}	1000	1000	1000	1000	1000
$S_{test}^{circle-b}$	416	1000	1000	1000	1000
S_{test}^{real-b}	1000	1000	1000	1000	1000
$S_{test}^{occlusion-b}$	435	556	809	972	1000
$S_{test}^{detection-b}$	22	30	44	56	15
$S_{test}^{probability-b}$	843	962	934	519	70

5.2 Training Details of the Shape Denoising Methods

5.2.1 Active Shape Model

We use 109 ground truth images from $S_{train}^{clean-h}$ to train the ASM model. For each image, we have 96 aligned points along the object boundary($N = 96$). After performing PCA, we keep the 50 largest eigenvectors ($p = 50$).

(a) 1st iteration (b) 2nd iteration (c) 3rd iteration

Figure 5.2: Example of finding the PCA shape.

The results for all six types of noise are showed in Table 5.2. It shows that ASM does not perform well on all the noise types.

Table 5.2: Performance(IoU) of ASM on the test set S_{test}^{all-h}.

IoU(begin) / Noise type	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	0.473	0.564	0.616	0.629	0.653
circle noise	0.558	0.600	0.645	0.667	0.667
real image noise	0.498	0.566	0.616	0.645	0.658
occlusion noise	0.285	0.361	0.467	0.559	0.643
detection image noise	0.288	0.486	0.626	0.637	0.630
thresholded probability noise	0.438	0.507	0.577	0.622	0.674

(a) input with salt and pepper noise (b) denoised result

Figure 5.3: Denoising example using the DBM.

5.2.2 Deep Boltzmann Machine

We use all the 159 images in $S_{train}^{clean-h}$ to train the DBM model. Since the size of the input image is 128×128, the number of units in the visible layer is 16384. We use two hidden layers in our DBM model.

Hidden layer size. We try different sizes for the two hidden layers. The results are shown in Table 5.3. We try three different sizes for the two hidden layers: 900, 2500 and 6400. From Table 5.3 one could see that the results are close between 900 and 2500, here we use the size 2500 as our choice. An example of a noisy input shape and the denoising result obtained using the DBM is shown in Figure 5.3.

Table 5.3: Performance (IoU) of the DBM on the test set S_{test}^{all-h} for different hidden layer sizes.

Noise type	IoU(begin) / hidden layer size	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	900	**0.718**	**0.714**	0.714	0.717	0.711
	2500	0.677	0.704	**0.717**	**0.724**	**0.720**
	6400	0.618	0.619	0.623	0.630	0.629
circle noise	900	**0.638**	**0.678**	**0.701**	**0.715**	0.717
	2500	0.598	0.644	0.685	0.714	**0.726**
	6400	0.595	0.615	0.623	0.632	0.636
real image noise	900	**0.684**	**0.700**	**0.713**	0.714	0.713
	2500	0.648	0.683	0.710	**0.718**	**0.721**
	6400	0.617	0.623	0.631	0.630	0.631
occlusion noise	900	**0.673**	**0.686**	**0.691**	**0.704**	0.711
	2500	0.636	0.655	0.665	0.687	**0.715**
	6400	0.590	0.601	0.605	0.620	0.628
detection noise	900	**0.638**	0.668	0.669	**0.703**	0.737
	2500	0.632	**0.673**	**0.672**	0.701	**0.742**
	6400	0.603	0.576	0.592	0.625	0.650
thresholded probability noise	900	**0.672**	**0.700**	**0.714**	0.708	0.724
	2500	0.623	0.664	0.701	**0.714**	**0.732**
	6400	0.597	0.614	0.625	0.623	0.643

5.2.3 Convolutional DBM

We use all the 159 images in $S_{train}^{clean-h}$ to train the CDBM model. The size of the visible layer is 128×128. Since we use nine 7×7 size filters to perform the convolution, the size of the first hidden layer is $9 \times 122 \times 122$.

Size of the second hidden layer. We try different sizes for the second hidden layer in CDBM. The results are shown in Table 5.4. The performance is the best when the size of the second layer is 2000. An example of a noisy input shape and the denoising result obtained using the CDBM is shown in Figure 5.4.

5.2.4 Energy-Based Prior Model

We use the five training sets $S_{train}^{salt-h}, S_{train}^{circle-h}, S_{train}^{real-h}, S_{train}^{occlusion-h}, S_{train}^{probability-h}$ as input to train the negative energy model and the generator. We define the negative energy model as a perceptron

Table 5.4: Performance (IoU) of CDBM on the test set S_{test}^{all-h}.

Noise type	IoU(begin) / 2nd layer size	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	400	0.704	0.774	0.812	0.822	0.817
	1000	0.776	0.832	0.868	0.884	0.884
	2000	**0.833**	**0.873**	**0.893**	**0.896**	**0.889**
circle noise	400	0.537	0.603	0.669	0.735	0.810
	1000	0.565	0.643	0.712	0.785	0.873
	2000	**0.588**	**0.657**	**0.723**	**0.797**	**0.882**
real image noise	400	0.529	0.609	0.682	0.745	0.804
	1000	0.635	**0.712**	**0.777**	**0.831**	**0.877**
	2000	**0.640**	**0.712**	0.773	0.822	0.836
occlusion noise	400	0.532	0.578	0.633	0.703	0.785
	1000	0.490	0.580	0.671	0.764	0.854
	2000	**0.630**	**0.670**	**0.715**	**0.775**	**0.858**
detection image noise	400	0.503	0.589	0.658	0.739	0.795
	1000	0.520	0.615	0.713	0.800	0.861
	2000	**0.617**	**0.656**	**0.728**	**0.810**	**0.869**
thresholded probability noise	400	0.553	0.630	0.695	0.747	0.793
	1000	0.604	0.684	0.754	0.812	0.854
	2000	**0.640**	**0.712**	**0.773**	**0.822**	**0.863**

with three fully connected layers. The generator model is composed of six transposed convolutional layers.

Size of the latent vector. We try different sizes for the latent vector. The results are shown in Table 5.5. We obtain best result when the length of the latent vector is 16000.

Choice of training data. We set the length of the latent vector as 16000. Then we try to use clean images set $S_{train}^{clean-h}$ as input instead of noisy images to train the negative energy model and the generator. The test results are shown in Table 5.6. It shows that the performance is better after using clean images as input. But if we add some data augmentation, e.g. rotation and scaling, on the noisy images, the results are the best.

Examples of a noisy input shapes and the denoising results obtained using the EBM are shown in Figure 5.5.

(a) input with salt and pepper noise (b) denoised result

Figure 5.4: Denoising example using CDBM.

5.2.5 U-Net

We use the five training sets $S_{train}^{salt-h}, S_{train}^{circle-h}, S_{train}^{real-h}, S_{train}^{occlusion-h}, S_{train}^{probability-h}$ as input to train the U-Net.

U-Net Architecture. We experimented with different architectures for the U-Net. We used bilinear interpolation instead of transposed convolution to complete the upsampling operation, added a batch normalization layer between convolution layers and ReLU. The U-Net has the following parameters: nBlocks is the number of blocks in U-Net, nConv is the number of convolutional layers in each block, and nFilter is the number of filters in the first convolutional layer. Table 5.7 shows all the results for each noise type with different combinations of these parameters. We obtain the best result when using four blocks, two convolutional layers in each block, and 128 filters in the first layer.

No skip connections. The U-Net architecture allows low-level information to shortcut across the network by using skip connections. We want to know how much do these skip connections contribute to the modeling of the object shape. We design an encoder-decoder network with the same architecture as the U-Net in our experiment without skip connections and retrain the network. Table 5.8 shows that all the performances are much worse without the skip connections.

5.2.6 DeepLabv3+

We use ResNet-101 as the model backbone and a pretrained model as initialization. The model is fine-tuned using the $S_{train}^{salt-h}, S_{train}^{circle-h}, S_{train}^{real-h}, S_{train}^{occlusion-h}, S_{train}^{probability-h}$ datasets. The results for each noise type are displayed in Table 5.9.

Table 5.5: Performance (IoU) of the EBM on test set S_{test}^{all-h} using different latent vector sizes.

Noise type	IoU(begin) / vector size	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	1000	0.785	0.780	0.783	0.786	0.782
	4000	0.789	0.784	**0.786**	0.788	**0.785**
	16000	**0.790**	**0.786**	**0.786**	**0.789**	**0.785**
circle noise	1000	0.667	0.708	0.742	**0.768**	0.785
	4000	0.653	0.706	0.741	**0.768**	0.787
	16000	**0.658**	**0.707**	**0.742**	**0.768**	**0.790**
real image noise	1000	0.753	0.766	0.778	0.780	0.783
	4000	**0.754**	**0.769**	0.781	0.784	0.786
	16000	0.752	0.768	**0.782**	**0.786**	**0.788**
occlusion noise	1000	**0.693**	0.711	**0.723**	0.741	0.774
	4000	0.692	**0.712**	**0.723**	**0.743**	0.775
	16000	0.689	0.709	0.721	0.742	**0.777**
detection image noise	1000	**0.736**	0.720	0.709	0.762	0.801
	4000	0.718	**0.729**	0.709	0.763	0.802
	16000	0.642	0.686	**0.719**	**0.767**	**0.811**
thresholded probability noise	1000	0.703	**0.739**	0.763	0.773	0.792
	4000	0.702	0.738	0.763	**0.776**	**0.797**
	16000	**0.703**	0.736	**0.764**	0.775	**0.797**

5.2.7 MAE

We use ViT-Base (Dosovitskiy et al., 2020) as the encoder backbone and a four-layer transformer as the decoder. The model is trained using the $S_{train}^{salt-h}, S_{train}^{circle-h}, S_{train}^{real-h}, S_{train}^{occlusion-h}, S_{train}^{probability-h}$ datasets with data augmentation (random sized cropping). The results for each noise type are displayed in Table 5.10.

5.2.8 DISSM

Decoder. Aligned images of size 256×256 are generated with the method described in section 3.1, forming the training set $S_{train-256}^{clean-h}$. We use MLP as the decoder, try $S_{train}^{clean-h}$ and $S_{train-256}^{clean-h}$ as the training data. We also try different architectures of decoder and compute the IoU of reconstructed images and original images. The results are displayed in Table 5.11.

Encoder. The Encoder is trained using the $S_{train}^{salt-h}, S_{train}^{circle-h}, S_{train}^{real-h}, S_{train}^{occlusion-h}, S_{train}^{probability-h}$ datasets. We try CNN and transformer as the encoder, the results for each noise type are displayed

Table 5.6: Performance(IoU) of EBM with different inputs on the test set S_{test}^{all-h}.

Noise type	IoU(begin) / Input	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	noisy input	0.790	0.786	0.786	0.789	0.785
	noise(rot and scale)	**0.883**	**0.885**	**0.890**	**0.894**	**0.895**
	clean input	0.874	0.874	0.877	0.882	0.880
circle noise	noisy input	**0.658**	**0.707**	0.742	0.768	0.790
	noise(rot and scale)	0.649	**0.707**	**0.767**	**0.826**	**0.885**
	clean input	0.579	0.661	0.737	0.810	0.874
real image noise	noisy input	0.752	0.768	0.782	0.786	0.788
	noise(rot and scale)	**0.804**	**0.840**	**0.870**	**0886**	**0.896**
	clean input	0.709	0.787	0.841	0.866	0.880
occlusion noise	noisy input	0.689	0.709	0.721	0.742	0.777
	noise(rot and scale)	**0.694**	**0.722**	**0.750**	**0.798**	**0.870**
	clean input	0.488	0.589	0.685	0.772	0.856
detection image noise	noisy input	0.642	0.686	0.719	0.767	0.811
	noise(rot and scale)	**0.699**	**0.730**	**0.771**	**0.838**	0.888
	clean input	0.547	0.648	0.767	0.829	**0.895**
thresholded probability noise	noisy input	0.703	0.736	0.764	0.775	0.797
	noise(rot and scale)	**0.712**	**0.772**	**0.822**	0.857	0.885
	clean input	0.646	0.744	0.815	**0.858**	**0.887**

in Table 5.12.

5.2.9 PCA-UNet

Decoder. We compute one PCA on grids of 16×16 patches on $S_{train}^{clean-h}$. This PCA was trained using Online PCA and 100 data augmentation perturbations per image. The data augmentation perturbations were: random resizing the 128×128 input in the interval $[120, 136]$, random rotation up to ± 15 degrees, and random cropping a 128×128 image with padding up to 16 pixels.

Encoder. We use the training sets $S_{train}^{salt-h}, S_{train}^{circle-h}, S_{train}^{real-h}, S_{train}^{occlusion-h}, S_{train}^{probability-h}$ to train the PCA-UNet. Data augmentation for the the training set includes random resizing to (120, 136), random rotation (degree=15) and random cropping (output size is 128, padding size on each border of the image is 8, pads with the value 0). The backbone of the encoder is ResNet50 (remove layer4), the number of PCA components is 64.

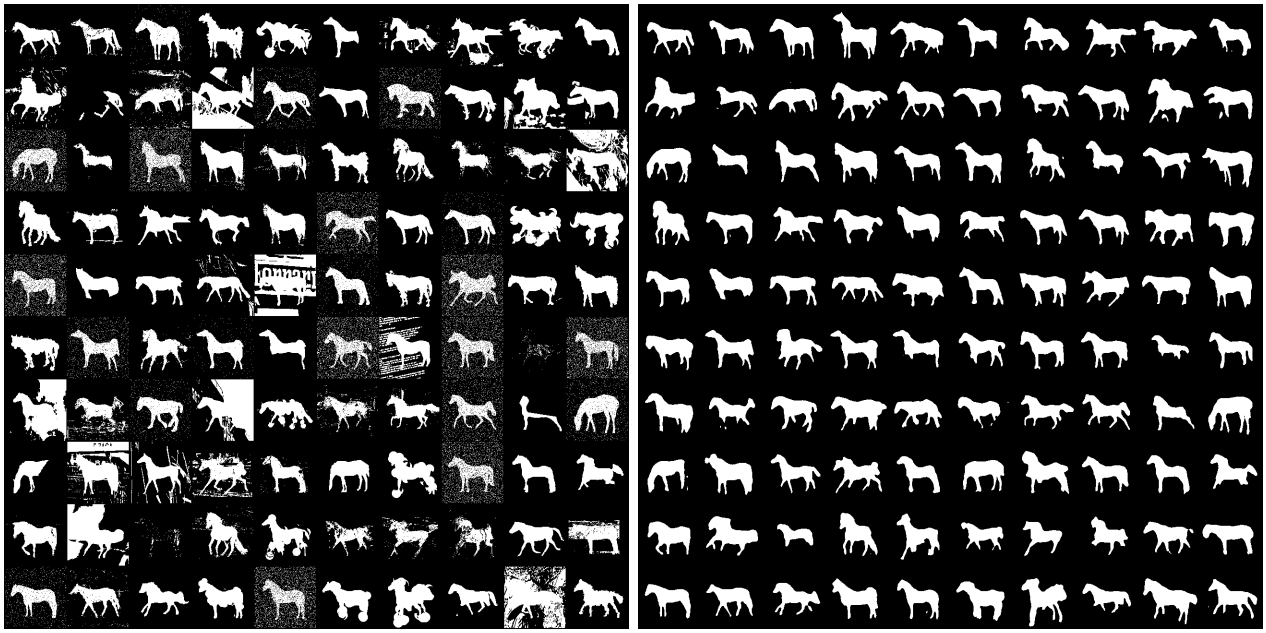


Figure 5.5: Denoising results using EBM. Left: noisy input shapes. Right: denoising results.

5.3 Results

5.3.1 Salt and Pepper Noise

An example of denoising results obtained with the methods evaluated on a shape corrupted by salt and pepper noise is shown in Fig. 5.6.

The Horse Dataset. The comparison of all methods on the horse dataset is displayed in Table 5.13 and Fig. 5.7 (top left). It is clear that U-Net and MAE are the best performers for handling salt and pepper noise, achieving average IoUs of 0.981 and 0.980, respectively. U-Net excels particularly in the 0.5-0.8 IoU range, showcasing its ability to maintain shape accuracy even under moderate noise conditions. On the other hand, MAE demonstrates the best performance in the higher IoU range (0.8-1), reflecting its superior shape recovery capability when the noise level is lower.

PCA-UNet follows closely with an average IoU of 0.960. Although it slightly underperforms compared to U-Net and MAE, PCA-UNet still demonstrates strong denoising capabilities, indicating its competitive performance in handling salt and pepper noise. Deeplabv3+ comes next with

Table 5.7: Performance(IoU) of U-Net on the test set S_{test}^{all-h}.

	Architecture			IoU(begin)				
Noise type	nBlocks	nConv	nFilter	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	4	2	64	0.960	0.971	0.980	0.985	0.988
	5	2	64	0.961	0.972	0.980	0.986	0.989
	4	3	64	0.961	0.971	0.979	0.985	0.987
	4	2	128	**0.966**	**0.976**	**0.983**	**0.988**	**0.992**
circle noise	4	2	64	0.800	0.844	0.877	0.897	0.947
	5	2	64	0.798	0.850	0.884	0.904	0.948
	4	3	64	**0.822**	**0.869**	**0.901**	**0.917**	0.949
	4	2	128	0.818	0.865	0.897	0.913	**0.952**
real image noise	4	2	64	0.925	0.947	0.965	0.977	0.987
	5	2	64	0.928	0.951	0.967	0.979	0.988
	4	3	64	**0.933**	0.953	0.968	0.978	0.986
	4	2	128	**0.933**	**0.954**	**0.970**	**0.982**	**0.990**
occlusion noise	4	2	64	0.839	0.864	0.873	0.898	0.955
	5	2	64	0.843	0.864	0.876	0.900	0.956
	4	3	64	**0.854**	**0.877**	**0.886**	**0.909**	0.958
	4	2	128	0.849	0.875	0.885	**0.909**	**0.961**
detection image noise	4	2	64	0.836	0.800	0.802	0.877	0.931
	5	2	64	**0.840**	0.802	**0.806**	0.880	**0.932**
	4	3	64	0.830	**0.805**	0.805	**0.881**	**0.932**
	4	2	128	0.833	0.797	0.805	0.878	0.930
thresholded probability noise	4	2	64	0.828	0.852	0.880	0.911	**0.940**
	5	2	64	0.831	0.859	0.884	0.913	**0.940**
	4	3	64	0.834	0.858	0.885	0.913	0.939
	4	2	128	**0.839**	**0.865**	**0.888**	**0.915**	0.939

an average IoU of 0.939, showing respectable performance but lagging slightly behind the top three methods.

EBM and CDBM provide intermediate results, with average IoUs of 0.889 and 0.877, respectively. Both methods manage to output complete shapes when the input images contain high-level noise, but they struggle to improve results when the noise level is lower, indicating that their performance plateaus under less noisy conditions.

DISSM and DBM, have similar performance, with average IoUs of 0.735 and 0.708.

The traditional Active Shape Model (ASM) exhibits the weakest performance, with an average IoU of 0.587. ASM significantly underperforms compared to other methods.

Table 5.8: Performance(IoU) of encoder-decoder on the test set S_{test}^{all-h}.

	Architecture			IoU(begin)				
Noise type	nBlocks	nConv	nFilter	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	4	2	64	0.864	0.864	0.865	0.866	0.864
	5	2	64	0.777	0.775	0.774	0.778	0.776
	4	3	64	0.855	0.856	0.856	0.859	0.858
	4	2	128	**0.877**	**0.874**	**0.876**	**0.879**	**0.877**
circle noise	4	2	64	0.696	0.737	0.782	0.825	0.860
	5	2	64	0.663	0.705	0.739	0.768	0.779
	4	3	64	0.698	0.740	0.784	0.824	0.855
	4	2	128	**0.707**	**0.750**	**0.796**	**0.839**	**0.874**
real image noise	4	2	64	0.812	0.829	0.843	0.852	0.862
	5	2	64	0.719	0.731	0.736	0.750	0.769
	4	3	64	0.806	0.823	0.839	0.846	0.855
	4	2	128	**0.822**	**0.840**	**0.856**	**0.865**	**0.874**
occlusion noise	4	2	64	**0.805**	**0.814**	**0.825**	**0.833**	0.843
	5	2	64	0.757	0.768	0.778	0.779	0.783
	4	3	64	0.754	0.771	0.780	0.803	0.841
	4	2	128	0.764	0.784	0.794	0.816	**0.858**
detection image noise	4	2	64	0.724	0.758	0.773	0.825	0.869
	5	2	64	**0.737**	0.728	0.697	0.749	0.799
	4	3	64	0.723	**0.761**	0.770	0.823	0.865
	4	2	128	0.717	0.756	**0.780**	**0.829**	**0.876**
thresholded probability noise	4	2	64	0.775	0.805	0.828	0.845	0.863
	5	2	64	0.726	0.750	0.765	0.772	0.797
	4	3	64	0.778	0.807	0.826	0.840	0.860
	4	2	128	**0.780**	**0.813**	**0.838**	**0.857**	**0.875**

The Bird Dataset. The comparison of all methods on the bird dataset is displayed in Table 5.14 and Fig. 5.8 (top left). It is clear that U-Net and MAE are the best performers for handling salt and pepper noise, achieving average IoUs of 0.985 and 0.987, respectively. PCA-UNet follows closely with an average IoU of 0.962.

5.3.2 Circle Noise

An example of denoising results obtained with the methods evaluated on a shape corrupted by circle noise is shown in Fig. 5.9.

Table 5.9: Performance (IoU) of DeepLabv3+ on the test set S_{test}^{all-h}.

IoU(begin) / Noise type	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	0.926	0.935	0.941	0.946	0.948
circle noise	0.743	0.795	0.834	0.872	0.926
real image noise	0.888	0.902	0.917	0.929	0.944
occlusion noise	0.793	0.815	0.829	0.853	0.905
detection image noise	0.567	0.624	0.755	0.850	0.917
thresholded probability noise	0.597	0.681	0.763	0.852	0.906

Table 5.10: Performance (IoU) of MAE on the test set S_{test}^{all-h}.

IoU(begin) / Noise type	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	0.963	0.973	0.982	0.989	0.996
circle noise	0.848	0.888	0.914	0.930	0.963
real image noise	0.942	0.963	0.978	0.987	0.995
occlusion noise	0.852	0.874	0.886	0.914	0.964
detection image noise	0.812	0.803	0.822	0.878	0.931
thresholded probability noise	0.845	0.868	0.893	0.917	0.941

The Horse Dataset. The comparison of all methods on the horse dataset is shown in Table 5.13 and Figure 5.7 (top right). Circle noise is more challenging than salt and pepper noise, especially when the noise level is high.

It is clear that PCA-UNet and MAE are the best performers for handling circle noise, achieving average IoUs of 0.922. PCA-UNet excels particularly in the 0.5-0.8 IoU range, showcasing its ability to maintain shape accuracy even under moderate noise conditions. On the other hand, MAE demonstrates the best performance in the higher IoU range (0.8-1), reflecting its superior shape recovery capability when the noise level is lower.

U-Net follows closely with an average IoU of 0.905, performing well across all noise levels, particularly in the higher IoU ranges where it achieves 0.952. U-Net's performance remains competitive and demonstrates strong shape recovery capabilities against circle noise.

Deeplabv3+ follows with an average IoU of 0.855.

Table 5.11: Performance(IoU) of MLP decoders with different architectures.

image size	lv size	num of layers	num of features	train(IoU)	test(IoU)
128×128	128	7	256	0.887	0.885
	64	7	256	0.916	0.921
	32	7	256	0.911	0.885
	32	7	128	0.906	0.861
	32	7	128	0.906	0.861
	32	6	256	0.874	0.836
256×256	128	7	256	0.944	0.933
	64	7	256	0.910	0.904
	64	5	128	0.944	0.930
	64	4	128	0.942	0.930
	64	**3**	**128**	**0.936**	**0.927**

EBM and CDBM provide intermediate performances, with average IoUs of 0.793 and 0.761, respectively. Both methods show decent performance, but they struggle to handle higher levels of noise.

DISSM and DBM underperform compared to the leading methods, with average IoUs of 0.723 and 0.690, respectively. The traditional Active Shape Model (ASM) performs the worst, with an average IoU of 0.643.

The Bird Dataset. The comparison of all methods on the bird dataset is shown in Table 5.14 and Figure 5.8 (top right). MAE and PCA-UNet are the best performers for handling circle noise, achieving average IoUs of 0.902 and 0.896. U-Net follows closely with an average IoU of 0.860.

5.3.3 Real Image Noise

An example of denoising results obtained by the methods evaluated on a shape corrupted by real image noise is shown in Fig. 5.10. The comparison of all methods is displayed in Table 5.15 and Fig. 5.11 (left). Real image noise is more challenging than salt and pepper noise, but easier than circle noise.

The Horse Dataset. MAE stands out as the top performer when dealing with real image noise, achieving the highest average IoU of 0.973. MAE consistently outperforms other methods across all noise levels, particularly in the 0.9-1 IoU range, where it achieves an impressive 0.995.

Table 5.12: Performance(IoU) of DISSM on the test set S_{test}^{all-h}.

		IoU(begin)				
Noise type	Architecture	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
salt and pepper noise	CNN	0.733	0.725	0.725	0.728	0.723
	Transformer	0.743	0.734	0.734	0.735	0.730
circle noise	CNN	0.681	0.709	0.719	0.727	0.730
	Transformer	0.674	0.710	0.720	0.731	0.735
real image noise	CNN	0.718	0.720	0.725	0.727	0.726
	Transformer	0.721	0.727	0.730	0.730	0.734
occlusion noise	CNN	0.701	0.712	0.708	0.715	0.723
	Transformer	0.713	0.719	0.719	0.724	0.729
detection image noise	CNN	0.709	0.684	0.684	0.718	0.754
	Transformer	0.718	0.666	0.686	0.722	0.753
thresholded probability noise	CNN	0.703	0.718	0.728	0.724	0.742
	Transformer	0.704	0.722	0.733	0.734	0.746

U-Net follows closely, with an average IoU of 0.966. It also demonstrates strong performance across all IoU ranges, achieving 0.990 in the 0.9-1 range, indicating its excellent shape recovery capabilities in real image noise scenarios.

PCA-UNet also performs very well, achieving an average IoU of 0.949. It is still a highly competitive option for handling real image noise.

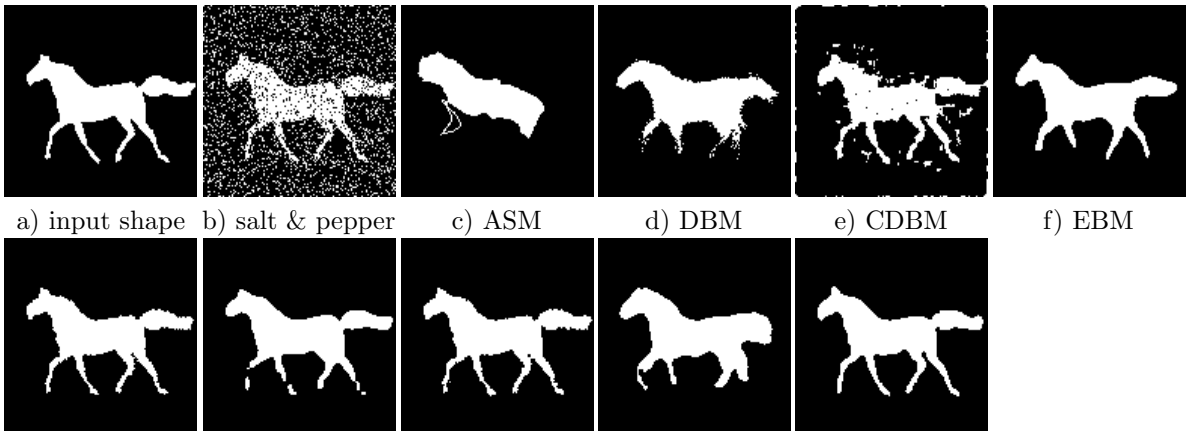
Deeplabv3+ performs moderately well with an average IoU of 0.908

EBM and CDBM provide intermediate performance, with average IoUs of 0.859 and 0.780, respectively.

DISSM and DBM underperform relative to the other methods, with average IoUs of 0.728 and 0.696, respectively. ASM performs the worst, with an average IoU of 0.597.

The Bird Dataset. MAE and U-Net stand out as the top performers when dealing with real image noise, achieving the highest average IoU of 0.971 and 0.970. PCA-UNet also performs very well, achieving an average IoU of 0.943.

If we categorize the input images by the percent of the object boundary that is left intact instead of the IoU, the comparison results are also displayed in Table 5.15.



a) input shape b) salt & pepper c) ASM d) DBM e) CDBM f) EBM

g) U-Net h) Deeplabv3+ i) MAE j) DISSM k) PCA-UNet

Figure 5.6: Example of results of different methods on a shape perturbed by salt and pepper noise.

5.3.4 Occlusion Noise

An example of denoising results obtained by the methods evaluated on a shape corrupted by occlusion noise is shown in Fig. 5.12. The comparison of all methods is displayed in Table 5.16 and Fig. 5.11 (right).

The Horse Dataset. MAE and U-Net are the top performers when dealing with occlusion noise. MAE achieves an average IoU of 0.917, performing particularly well in the 0.8-1 range.

MAE and U-Net are the best performers for handling occlusion noise, achieving average IoUs of 0.917 and 0.915, respectively. MAE performs particularly well in the 0.8-1 range.

PCA-UNet also performs very well, with an average IoU of 0.910. It is the best performer in the 0.5-0.8 range (IoU), showcasing its ability to maintain shape accuracy even under moderate noise conditions.

Deeplabv3+ delivers moderate performance with an average IoU of 0.858.

EBM and CDBM provide intermediate performances, achieving average IoUs of 0.798 and 0.769, respectively.

DISSM and DBM underperform relative to the other methods, with average IoUs of 0.724 and 0.685, respectively. ASM performs the worst, with an average IoU of 0.522.

Table 5.13: Performance (IoU) of methods evaluated on S_{test}^{salt-h}, $S_{test}^{circle-h}$, $S_{test}^{detection-h}$, $S_{test}^{probability-h}$.

Method \ IoU	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	avg	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	avg
	Salt and Pepper Noise						Circle Noise					
ASM	0.476	0.564	0.616	0.629	0.653	0.587	0.558	0.600	0.645	0.667	0.667	0.643
DBM	0.677	0.704	0.717	0.724	0.720	0.708	0.598	0.644	0.685	0.714	0.726	0.690
CDBM	0.833	0.873	0.893	0.896	0.889	0.877	0.588	0.657	0.723	0.797	0.882	0.761
EBM	0.883	0.885	0.890	0.894	0.895	0.889	0.649	0.707	0.767	0.826	0.885	0.793
U-Net	**0.966**	**0.976**	**0.983**	0.988	0.992	**0.981**	0.818	0.865	0.897	0.913	0.952	0.905
Deeplabv3+	0.926	0.935	0.941	0.946	0.948	0.939	0.743	0.795	0.834	0.872	0.926	0.855
MAE	0.963	0.973	0.982	**0.989**	**0.996**	0.980	0.848	0.888	0.914	**0.930**	**0.963**	**0.922**
DISSM	0.743	0.734	0.734	0.735	0.730	0.735	0.674	0.710	0.720	0.731	0.735	0.723
PCA-UNet	0.953	0.957	0.962	0.964	0.963	0.960	**0.878**	**0.904**	**0.919**	0.928	0.942	**0.922**
	Detection Image Noise						Thresholded Probability Noise					
ASM	0.288	0.486	0.626	0.637	0.630	0.620	0.438	0.507	0.577	0.622	0.674	0.583
DBM	0.632	0.673	0.672	0.701	0.742	0.711	0.623	0.664	0.701	0.714	0.732	0.698
CDBM	0.617	0.656	0.728	0.810	0.869	0.815	0.640	0.712	0.773	0.822	0.863	0.781
EBM	0.699	0.730	0.771	0.838	0.888	0.844	0.712	0.772	0.822	0.857	0.885	0.825
U-Net	0.833	0.797	0.805	0.878	0.930	0.887	0.839	0.865	0.888	0.915	0.939	0.896
Deeplabv3+	0.567	0.624	0.755	0.850	0.917	0.852	0.597	0.681	0.763	0.852	0.906	0.786
MAE	0.812	0.803	0.822	0.878	**0.931**	0.888	0.845	0.868	0.893	**0.917**	**0.941**	0.900
DISSM	0.718	0.666	0.686	0.722	0.753	0.728	0.704	0.722	0.733	0.734	0.746	0.731
PCA-UNet	**0.835**	**0.821**	**0.831**	**0.881**	0.927	**0.890**	**0.863**	**0.881**	**0.899**	0.916	0.929	**0.903**

The Bird Dataset. U-Net is the top performers when dealing with occlusion noise, achieving an average of IoU of 0.894. MAE and PCA-UNet also perform very well, achieving average IoUs of 0.886 and 0.884.

5.3.5 Detection Image Noise

An example of denoising results obtained by the methods evaluated on a shape corrupted by detection image noise is shown in Fig. 5.13.

The detection image noise is the most challenging noise of all six noises. None of the methods performs well.

The Horse Dataset. The comparison of all methods on the horse dataset is displayed in Table 5.13 and Fig. 5.7 (bottom left). PCA-UNet emerges as the top performer, achieving an impressive average IoU of 0.890 for detection image noise. It excels across most IoU ranges. This

Table 5.14: Performance (IoU) of methods evaluated on S_{test}^{salt-b}, $S_{test}^{circle-b}$, $S_{test}^{detection-b}$, $S_{test}^{probability-b}$.

Method \ IoU	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	avg	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	avg
	Salt and Pepper Noise						Circle Noise					
DBM	0.627	0.635	0.643	0.645	0.656	0.641	0.434	0.556	0.626	0.645	0.649	0.602
CDBM	0.814	0.811	0.816	0.812	0.819	0.814	0.570	0.662	0.730	0.778	0.819	0.731
EBM	0.891	0.893	0.901	0.898	0.909	0.898	0.542	0.662	0.760	0.830	0.884	0.761
U-Net	**0.977**	**0.983**	**0.988**	0.987	0.988	0.985	0.685	0.799	0.871	0.906	0.938	0.860
Deeplabv3+	0.942	0.946	0.952	0.953	0.955	0.950	0.684	0.778	0.842	0.881	0.919	0.839
MAE	0.975	0.982	**0.988**	**0.992**	**0.996**	**0.987**	0.821	0.869	**0.906**	**0.926**	**0.942**	**0.902**
PCA-UNet	0.958	0.961	0.965	0.964	0.964	0.962	**0.840**	**0.874**	0.900	0.916	0.920	0.896
	Detection Image Noise						Thresholded Probability Noise					
DBM	0.518	0.590	0.590	0.653	0.671	0.609	0.573	0.587	0.604	0.627	0.609	0.595
CDBM	0.528	0.631	0.697	0.759	0.800	0.693	0.610	0.673	0.716	0.759	0.806	0.685
EBM	0.648	0.751	0.782	0.855	0.895	0.793	0.731	0.790	0.828	0.855	0.836	0.797
U-Net	0.723	**0.807**	**0.815**	0.877	**0.925**	**0.832**	0.849	0.874	**0.890**	**0.910**	**0.929**	**0.879**
Deeplabv3+	0.703	0.782	0.789	0.865	0.912	0.813	0.796	0.833	0.860	0.874	0.878	0.839
MAE	0.700	0.740	0.772	0.852	0.897	0.795	0.859	**0.878**	0.887	0.897	0.824	0.878
PCA-UNet	**0.744**	0.799	0.799	**0.878**	0.891	0.826	**0.862**	0.876	0.887	0.892	0.808	0.877

indicates PCA-UNet's ability to recover clean shapes even when the detection noise severely distorts the input.

MAE and U-Net follow closely with an average IoU of 0.888 and 0.887. MAE performs the best in the 0.9-1 range.

Deeplabv3+ provides a respectable performance with an average IoU of 0.852.

EBM and CDBM offer intermediate results, with average IoUs of 0.844 and 0.815, respectively.

DISSM and DBM show weaker performance, with average IoUs of 0.728 and 0.711, respectively. ASM performs the worst, with an average IoU of 0.620.

The Bird Dataset. The comparison of all methods on the bird dataset is displayed in Table 5.14 and Fig. 5.8 (bottom left). U-Net and PCA-UNet are the top performers, achieving average IoUs of 0.832 and 0.826.

5.3.6 Thresholded Probability Noise

An example of denoising results obtained by the methods evaluated on a shape corrupted by thresholded probability noise is shown in Fig. 5.14.

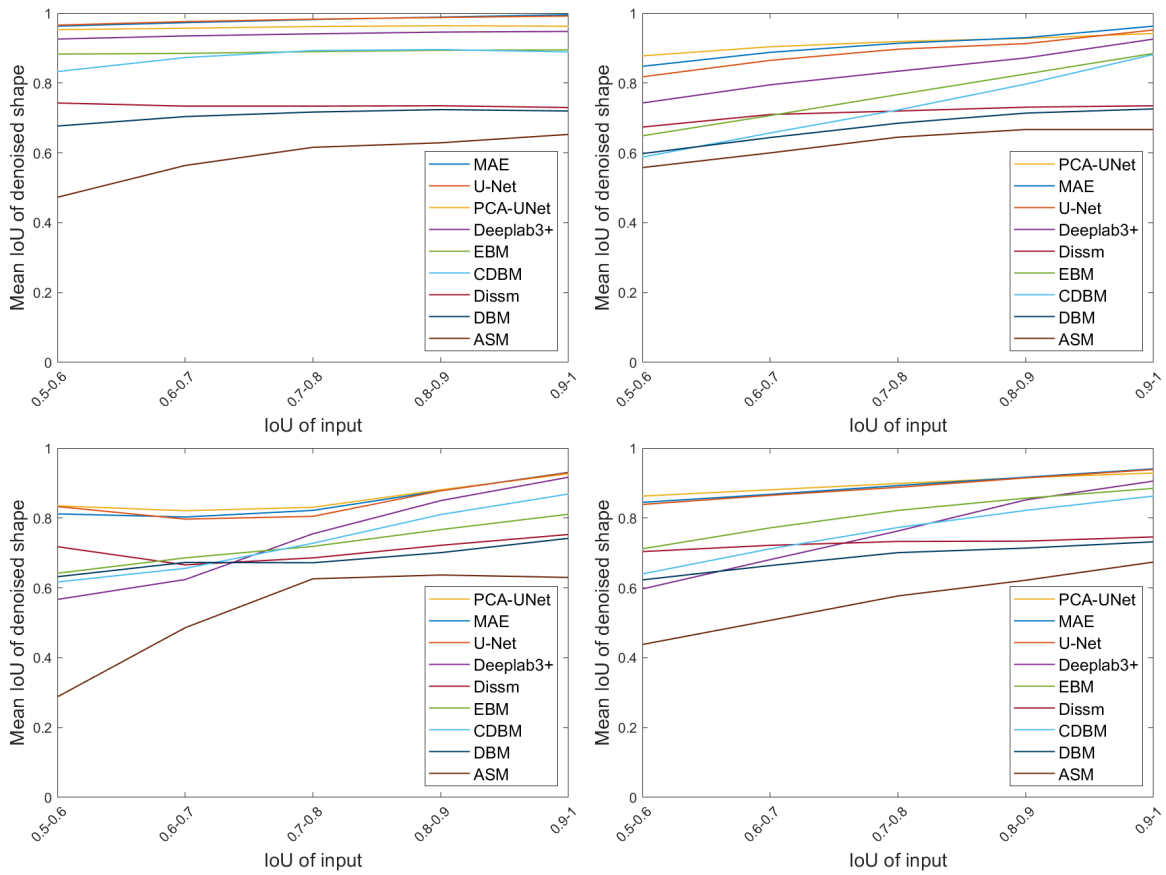


Figure 5.7: Performance on salt and pepper noise data S_{test}^{salt-h} (top left), circle noise data $S_{test}^{circle-h}$ (top right),detection noise $S_{test}^{detection-h}$ (bottom left) and thresholded probability noise $S_{test}^{probability-h}$ (bottom right).

The Horse Dataset. The comparison of all methods on the horse dataset is displayed in Table 5.13 and Fig. 5.7(bottom right). The thresholded probability noise is as challenging as the detection noise when the IoU of input noise image is above 0.8, but it is easier than the detection noise when the IoUs are below 0.8. PCA-UNet, MAE and U-Net are the top performers when dealing with thresholded probability noise, achieving average IoUs of 0.903, 0.900 and 0.896, respectively. PCA-UNet excels particularly in the 0.5-0.8 range, showcasing its ability to maintain shape accuracy even under moderate noise conditions. On the other hand, MAE demonstrates the best performance in the higher IoU range (0.8-1), reflecting its superior shape recovery capability when the noise level is lower.

EBM follows with an average IoU of 0.825.

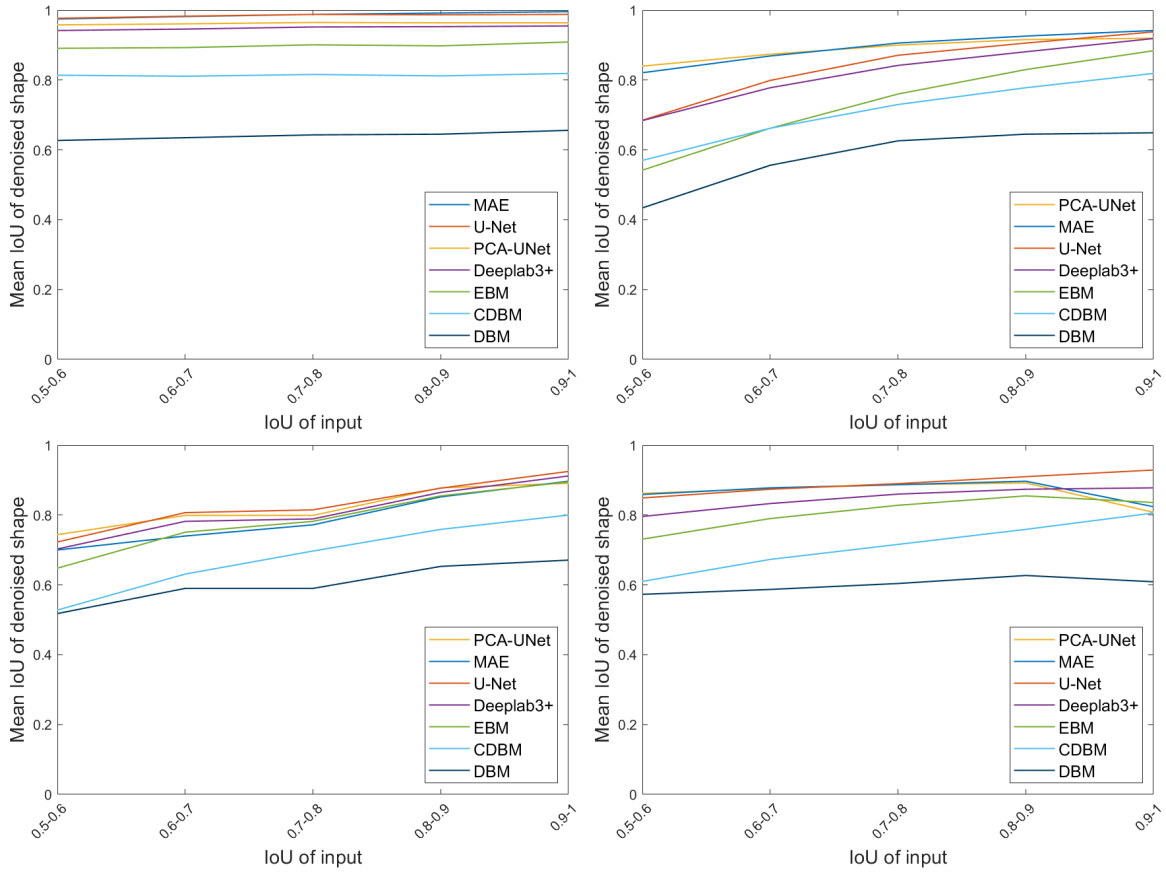
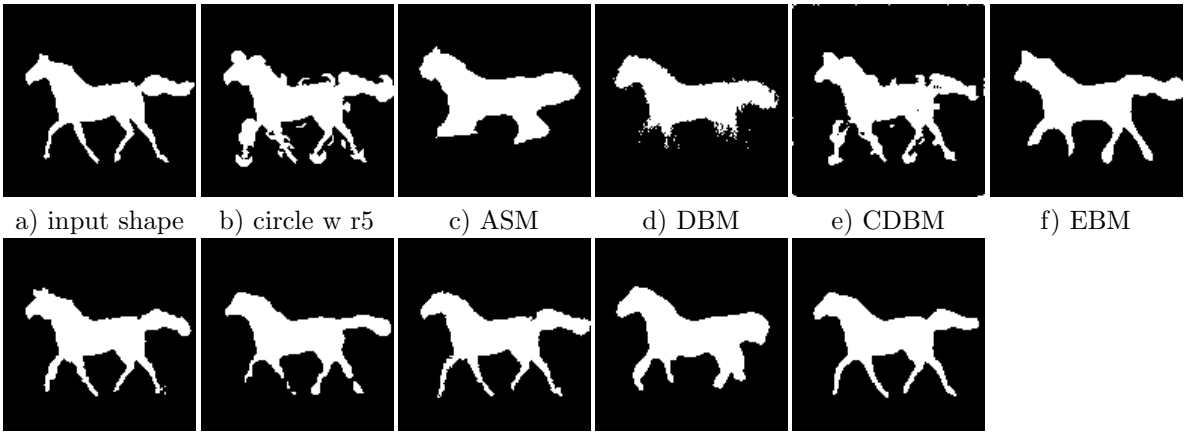


Figure 5.8: Performance on salt and pepper noise data S_{test}^{salt-b} (top left), circle noise data $S_{test}^{circle-b}$ (top right),detection noise $S_{test}^{detection-b}$ (bottom left) and thresholded probability noise $S_{test}^{probability-b}$ (bottom right).

Deeplabv3+ and CDBM provide intermediate results, with average IoUs of 0.786 and 0.781, respectively.

DISSM and DBM show weaker performance compared to the others, with average IoUs of 0.731 and 0.698, respectively. ASM performs the worst, with an average IoU of 0.583.

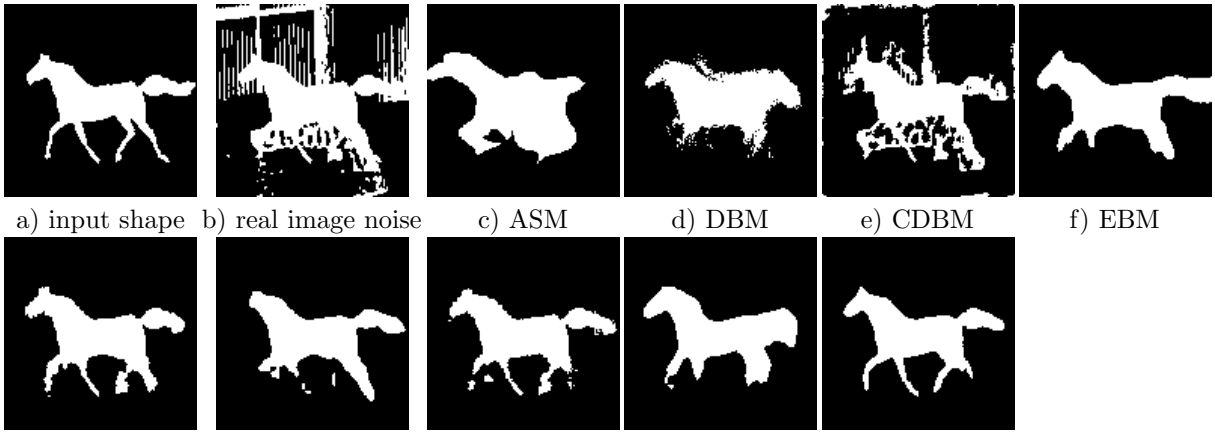
The Bird Dataset. The comparison of all methods on the bird dataset is displayed in Table 5.14 and Fig. 5.8(bottom right). U-Net MAE and PCA-UNet are the top performers, achieving average IoUs of 0.879, 0.878 and 0.877.



a) input shape b) circle w r5 c) ASM d) DBM e) CDBM f) EBM

g) U-Net h) Deeplabv3+ i) MAE j) DISSM k) PCA-UNet

Figure 5.9: Example of results of different methods on a shape perturbed by circle noise.



a) input shape b) real image noise c) ASM d) DBM e) CDBM f) EBM

g) U-Net h) Deeplabv3+ i) MAE j) DISSM k)PCA-UNet

Figure 5.10: Example of results of different methods on a shape perturbed by real image noise.

Table 5.15: Comparison of methods against real image noise.

Method	IoU(begin)					Ratio of remaining boundaries					avg
	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	
	S_{test}^{real-h}										
ASM	0.498	0.566	0.616	0.645	0.658	0.542	0.540	0.547	0.567	0.609	0.597
DBM	0.648	0.683	0.710	0.718	0.721	0.627	0.622	0.633	0.663	0.711	0.696
CDBM	0.660	0.730	0.790	0.839	0.881	0.601	0.629	0.657	0.702	0.813	0.780
EBM	0.804	0.840	0.870	0.886	0.896	0.713	0.738	0.765	0.815	0.882	0.859
U-Net	0.933	0.954	0.970	0.982	0.990	0.843	0.863	0.899	0.937	0.982	0.966
Deeplabv3+	0.868	0.890	0.910	0.922	0.934	0.774	0.805	0.835	0.873	0.922	0.908
MAE	**0.942**	**0.963**	**0.978**	**0.987**	**0.995**	0.851	0.882	**0.912**	**0.949**	**0.987**	**0.973**
DISSM	0.721	0.727	0.730	0.730	0.734	0.696	0.712	0.708	0.721	0.733	0.728
PCA-UNet	0.930	0.942	0.952	0.959	0.960	**0.854**	**0.885**	0.909	0.935	0.958	0.949
	S_{test}^{real-b}										
DBM	0.636	0.635	0.638	0.650	0.651	0.663	0.642	0.648	0.655	0.640	0.642
CDBM	0.770	0.785	0.792	0.807	0.811	0.735	0.739	0.758	0.774	0.798	0.793
EBM	0.852	0.873	0.881	0.895	0.901	0.773	0.780	0.819	0.852	0.888	0.880
U-Net	0.954	0.963	**0.971**	**0.976**	**0.985**	0.856	**0.883**	**0.920**	0.946	0.976	0.970
Deeplabv3+	0.915	0.925	0.932	0.939	0.944	0.805	0.840	0.887	0.910	0.937	0.931
MAE	**0.958**	**0.968**	**0.971**	0.974	0.983	0.864	0.870	0.916	**0.954**	**0.977**	**0.971**
PCA-UNet	0.934	0.941	0.945	0.945	0.952	**0.883**	0.869	0.911	0.931	0.947	0.943

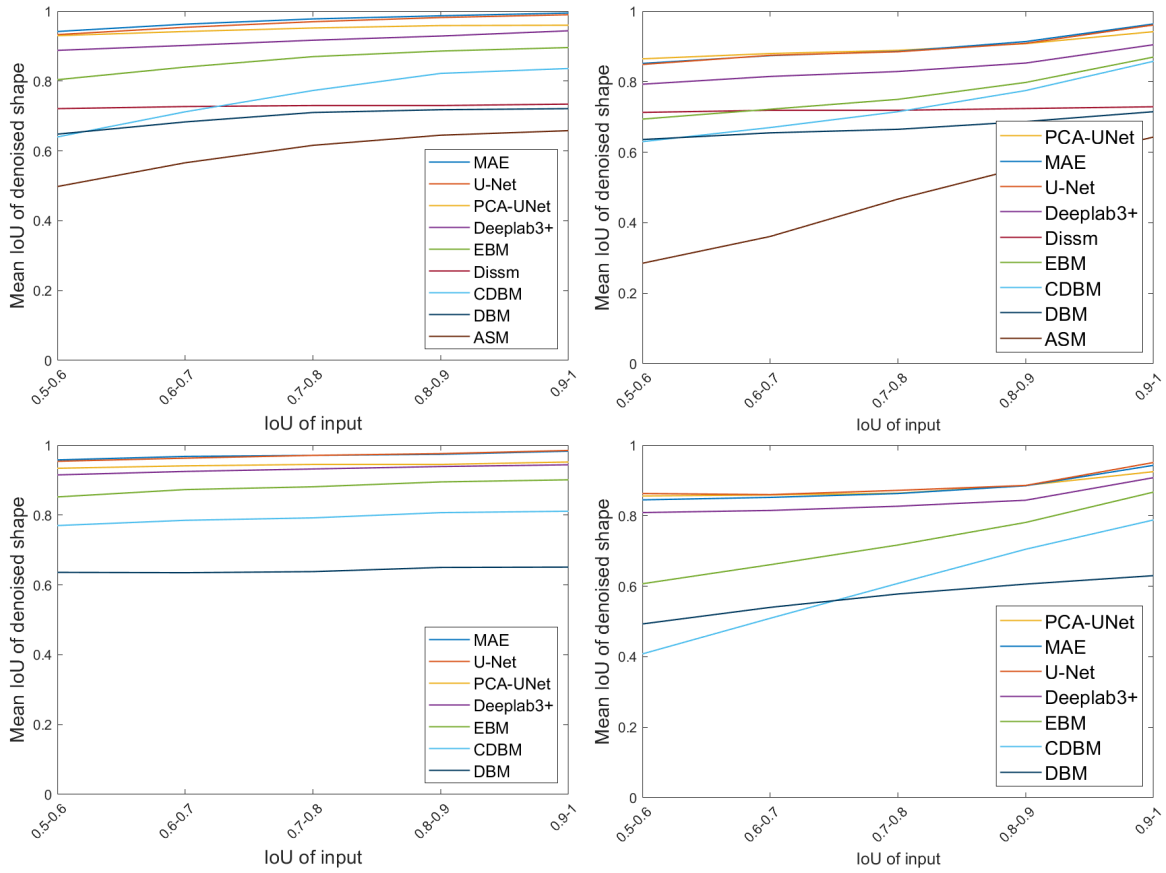
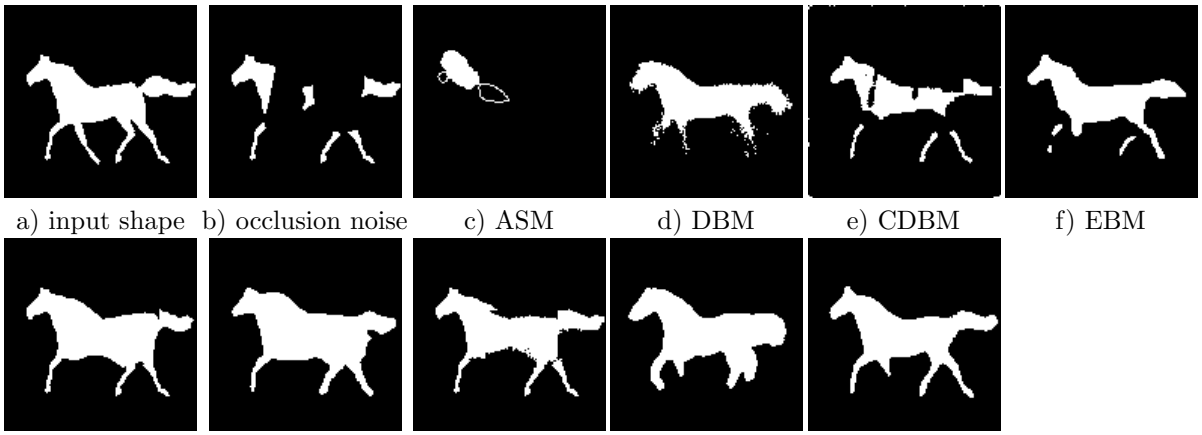


Figure 5.11: Performance on real image noise S_{test}^{real-h} (top left), S_{test}^{real-b} (bottom left) and occlusion noise $S_{test}^{occlusion-h}$ (top right), $S_{test}^{occlusion-b}$ (bottom right).



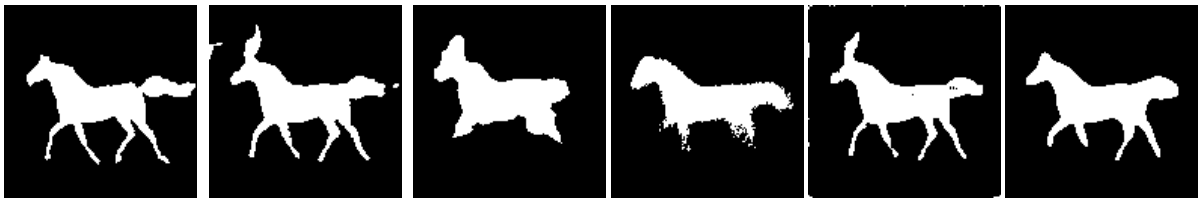
a) input shape b) occlusion noise c) ASM d) DBM e) CDBM f) EBM

g) U-Net h) Deeplabv3+ i) MAE j) DISSM k)PCA-UNet

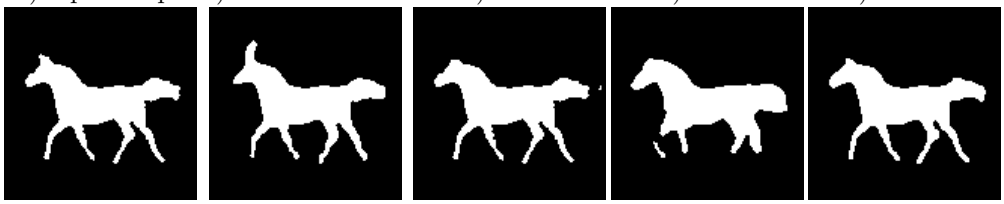
Figure 5.12: Example of results of different methods on a shape perturbed by occlusion noise.

Table 5.16: Comparison of methods against occlusion noise.

Method	IoU(begin)					Ratio of remaining boundaries					avg
	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	
	$S_{test}^{occlusion-h}$										
ASM	0.285	0.361	0.467	0.559	0.643	0.358	0.425	0.487	0.571	0.645	0.522
DBM	0.636	0.655	0.665	0.687	0.715	0.649	0.664	0.675	0.694	0.713	0.685
CDBM	0.630	0.670	0.715	0.775	0.858	0.659	0.696	0.740	0.799	0.863	0.769
EBM	0.694	0.722	0.750	0.798	0.870	0.710	0.740	0.773	0.820	0.876	0.798
U-Net	0.849	0.875	0.885	0.909	0.961	0.835	0.869	0.900	**0.938**	**0.974**	0.915
Deeplabv3+	0.793	0.815	0.829	0.853	0.905	0.794	0.818	0.842	0.875	0.913	0.858
MAE	0.852	0.874	0.886	**0.914**	**0.964**	0.835	0.871	**0.903**	0.941	0.977	**0.917**
DISSM	0.713	0.719	0.719	0.724	0.729	0.718	0.724	0.720	0.727	0.726	0.724
PCA-UNet	**0.865**	**0.880**	**0.889**	0.908	0.942	**0.847**	**0.878**	0.902	0.929	0.951	0.910
	$S_{test}^{occlusion-b}$										
DBM	0.493	0.540	0.578	0.606	0.630	0.545	0.567	0.584	0.599	0.628	0.584
CDBM	0.408	0.509	0.608	0.705	0.788	0.552	0.572	0.638	0.697	0.783	0.643
EBM	0.607	0.661	0.717	0.781	0.867	0.658	0.698	0.746	0.805	0.877	0.752
U-Net	**0.863**	**0.860**	**0.872**	**0.886**	**0.951**	0.820	**0.866**	**0.893**	**0.929**	**0.969**	**0.894**
Deeplabv3+	0.809	0.815	0.827	0.844	0.908	0.797	0.823	0.843	0.874	0.919	0.849
MAE	0.845	0.852	0.863	0.885	0.943	0.814	0.858	0.886	0.919	0.959	0.886
PCA-UNet	0.856	0.859	0.864	**0.886**	0.925	**0.829**	0.864	0.889	0.911	0.933	0.884

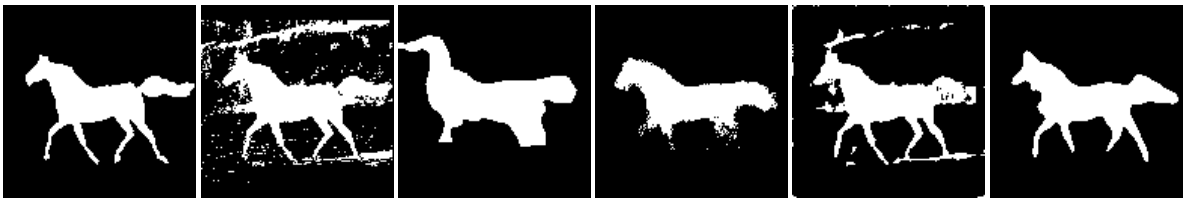


a) input shape b) detection noise c) ASM d) DBM e) CDBM f) EBM

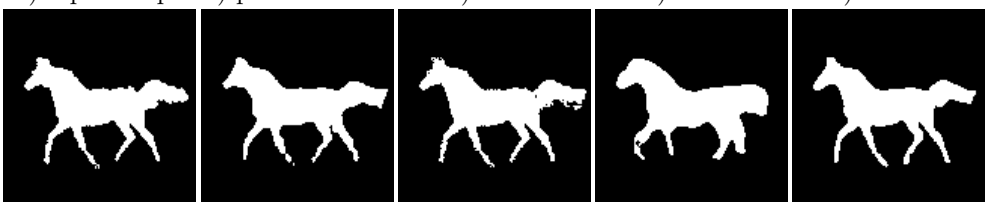


g) U-Net h) Deeplabv3+ i) MAE j) DISSM k)PCA-UNet

Figure 5.13: Example of results of different methods on a shape perturbed by detection image noise.



a) input shape b) prob. thr=10 c) ASM d) DBM e) CDBM f) EBM



g) U-Net h) Deeplabv3+ i) MAE j) DISSM k)PCA-UNet

Figure 5.14: Example of results of different methods on a shape perturbed by thresholded probability noise.

CHAPTER 6

SHAPE DENOISING IN THE WILD

In this chapter, we study shape denoising methods for denoising non-aligned shapes. The criterion we use to estimate the quality of the denoising result is the IoU (3.6).

6.1 Dataset Construction

‘

We use the 327 mask horse images with their original size in the Weizmann Horse dataset and 417 mask bird images in the Caltech-UCSD Bird dataset for our experiments.

(a) mask image1 (b) mask image2

Figure 6.1: Two original size mask images in the horse dataset.

From the 327 original size horse images, 159 images were randomly selected as the training set $S^{clean-h}_{train-original}$ and the other 168 images as the test set $S^{clean-h}_{test-original}$. Each image in $S^{clean-h}_{train-original}$ and $S^{clean-h}_{test-original}$ was padded with zeros to make it of size 560×560, obtaining a new training set $S^{clean-h}_{train-ori560}$ and a new test set $S^{clean-h}_{test-ori560}$.

From the 417 original size bird images, 207 images were randomly selected as the training set $S^{clean-b}_{train-original}$ and the other 210 images as the test set $S^{clean-b}_{test-original}$.

6.1.1 Noisy Image Datasets

Training Sets (unaligned). For each image in training set $S^{clean-h}_{train-original}$, noisy versions were generated as follows:

- For a training image with size $h \times w$ in $S^{clean-h}_{train-original}$, real image noise was generated by randomly selecting a $h \times w$ size random patch from a randomly selected image of the PASCAL VOC 2012 Dataset. If the randomly selected image is smaller than $h \times w$ size, we will resize the image. The patch was converted to grayscale, and different thresholds in $\{10, 40, 70, 100, 130, 160, 190, 220, 250\}$ were used on the grayscale patch to generate nine binary images. These binary images were used as the background of the selected training image. The ratio between the length of the remaining correct boundary after adding noise and the length of the original boundary was computed. The generated images with the remaining boundaries ratio above 0.5 were added to the real image noise training set $S^{real-h}_{train-original}$, which contains 20030 images.

- For a training image with size $h \times w$ in $S^{clean-h}_{train-original}$, occlusion noise was generated by randomly selecting a $h \times w$ size random patch from a randomly selected mask image of the PASCAL VOC 2012 Dataset. If the randomly selected image is smaller than $h \times w$ size, we will resize the image. The binary patch was used to occlude the foreground of the selected training image. The generated images with the remaining boundaries ratio above 0.5 are added to the occlusion noise training set $S^{occlusion-h}_{train-original}$, which contains 7382 images.

- thresholded probability noise sets are generated as described in Section 3.2.6. For each image in the training set $S^{clean-h}_{train-original}$, we use the corresponding original color image in the Weizmann Horse dataset to build a training set $S^{color-h}_{train}$. The pixels on each color image in $S^{color-h}_{train}$ are clustered into 800 clusters using k-means clustering. Based on the clustering result and the corresponding mask image in $S^{clean-h}_{train-original}$, we compute the probability that each pixel is in the object and generate a probability map of this image. For each grayscale probability map, we use different thresholds in $\{10/255, 20/255, 30/255, ..., 250/255\}$ to generate binary noise images. All the binary images build the thresholded probability noise set $S^{probability-h}_{train-original}$, which contains 3975 images.

For bird images in training set $S^{clean-b}_{train-original}$, use same methods to generate $S^{real-b}_{train-original}$ (13764 images), $S^{occlusion-b}_{train-original}$ (5175 images), and $S^{probability-b}_{train-original}$ (9126 images).

Training Sets (same size). Each horse image in training sets $S^{real-h}_{train-original}$, $S^{occlusion-h}_{train-original}$ and $S^{probability-h}_{train-original}$ was resized so that the length of the long size is 256, then the short side was padded to make its size as 256×256. The training sets with same sized images are $S^{real-h}_{train-256}$, $S^{occlusion-h}_{train-256}$ and $S^{probability-h}_{train-256}$. Also generated were training sets $S^{real-h}_{train-384}$, $S^{occlusion-h}_{train-384}$ and $S^{probability-h}_{train-384}$ using the same method with the bigger size 384.

For bird images in training set $S^{real-b}_{train-original}$, $S^{occlusion-b}_{train-original}$ and $S^{probability-b}_{train-original}$, generated $S^{real-b}_{train-256}$, $S^{occlusion-b}_{train-256}$, $S^{probability-b}_{train-256}$, $S^{real-b}_{train-384}$, $S^{occlusion-b}_{train-384}$ and $S^{probability-b}_{train-384}$.

Test Sets. For each horse image in test set $S^{clean-h}_{test-original}$, we use the above methods (used in unaligned part) to generate noisy images, obtaining real image noise $S^{real-h}_{test-original}$, occlusion noise $S^{occlusion-h}_{test-original}$ and thresholded probability noise set $S^{probability-h}_{test-original}$. We organize the noisy images by their IoU with the original image. For each IoU level (e.g. 0.5-0.6, 0.6-0.7, ...) we randomly select at most 1000 images for testing.

For each image in the test set $S^{clean-h}_{test-original}$, we use the same process described in Section 5.1.2 to generate detection image noise set $S^{detection-h}_{test-original}$.

We denote the set $\{S^{real-h}_{test-original}, S^{occlusion-h}_{test-original}, S^{detection-h}_{test-original}, S^{probability-h}_{test-original}\}$ as $S^{all-h}_{test-original}$. We also generated bird test set $\{S^{real-b}_{test-original}, S^{occlusion-b}_{test-original}, S^{detection-b}_{test-original}, S^{probability-b}_{test-original}\}$ and denote it as $S^{all-b}_{test-original}$ Table 6.1 shows the number of images in each category on $S^{all-h}_{test-original}$ and $S^{all-b}_{test-original}$.

Table 6.1: Number of noisy test images in each noise category.

IoU(begin) / Test set	0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1
$S^{real-h}_{test-original}$	5	117	548	1000	1000	1000	1000	1000	1000	1000
$S^{occlusion-h}_{test-original}$	0	7	164	434	719	999	1051	1101	1388	1877
$S^{detection-h}_{test-original}$	0	1	1	2	3	3	7	14	77	60
$S^{probability-h}_{test-original}$	17	24	39	70	129	334	697	1079	1324	486
$S^{real-b}_{test-original}$	921	1000	1000	1000	1000	1000	916	900	952	1000
$S^{occlusion-b}_{test-original}$	0	23	244	690	1000	1000	1000	1000	1000	1000
$S^{detection-b}_{test-original}$	4	5	8	11	16	28	31	41	52	14
$S^{probability-b}_{test-original}$	586	497	618	701	730	683	637	442	185	16

6.2 Shape Denoising Approaches

The shape denoising in the wild problem can be seen as a instance segmentation task that aims to detect and segment objects within a image. There are two common approaches to perform instance segmentation known as 'one-step' and 'two-step' methods.

One-step instance segmentation methods combine object detection and instance segmentation into a single unified process. These methods directly predict the corresponding pixel-level masks for each instance in a single forward pass through the neural network.

(a) real image noise (b) occlusion noise (c) detection noise (d) thresholded prob noise

Figure 6.2: Original size images with noise in horse dataset.

Two-step instance segmentation methods separate the task into two distinct steps: object proposal generation and instance segmentation. First, the algorithm generate a set of region proposals or candidate object bounding boxes using object detection techniques. Then, these regions are fed into a separate instance segmentation model, which performs pixel-wise segmentation for each detected region proposal.

6.2.1 One-Step

In one-step approaches, we will directly use the shape images as input to get the denoised results. Since the target shape varies in different positions and sizes within the image, we will add data augmentation on the input images to improve the model's generalization ability. The methods we will apply in this part include DISSM, U-Net, MAE and PCA-UNet.

6.2.2 Two-Step

In the two-step approaches, we first perform object detection on the images at their original sizes to obtain the position and size information of the targets. Based on this information, we then crop appropriate image patches, resize them to a consistent size, and proceed to the second step of shape denoising.

The object detection method we apply is Faster-RCNN (Ren et al., 2015), the shape denoising methods we apply are U-Net, MAE and PCA-UNet.

Faster-RCNN. First, we need to build a new dataset to train the Faster-RCNN we need. For each image in $S_{train}^{clean-h}$, its size is resized from 128 to a random number $s_{new} \in [102, 152]$. Next, we randomly choose two values $h_{before} \in [0, 152 - s_{new}], w_{before} \in [0, 152 - s_{new}]$, which represents the padding size to be applied before the top and left edges, respectively. Then we use the two values $h_{after} = 152 - s_{new} - h_{before}$ and $w_{after} = 152 - s_{new} - w_{before}$ as the padding size to be applied after the bottom and right edges, resulting in a final size of 152×152. All the padding values are 0. The new constructed training dataset $S_{train-random152}^{clean-h}$ contains 7101 images. We can compute the coordinates (x_1, y_1, x_2, y_2) as

$$x_1 = w_{before}, y_1 = h_{before}, x_2 = w_{before} + s_{new}, y_2 = h_{before} + s_{new} \tag{6.1}$$

The test dataset $S_{test-random152}^{clean-h}$ is generated with the same method, which contains 7622 images. For each image in $S_{train-random152}^{clean-h}$, three noisy versions are generated as follows:

- real image noise was generated by randomly selecting a 152×152 size random patch from a randomly selected image of the PASCAL VOC 2012 Dataset. The patch was converted to grayscale, then randomly choose a threshold in $\{10, 20, 30, ..., 250\}$ to generate a binary image. These binary images were used as the background of the selected training image. The generated images with the remaining boundaries ratio above 0.5 are added to the real image noise training set $S_{train-random152}^{real-h}$.
- occlusion noise was generated by randomly selecting a 152×152 size random patch from a randomly selected mask image of the PASCAL VOC 2012 Dataset. The binary patch was used to occlude the foreground of the selected training image. The generated images with the remaining boundaries ratio above 0.5 are added to the occlusion noise training set $S_{train-random152}^{occlusion-h}$.
- For each grayscale probability map with original size, we resize and pad it to be the size 152×152, and the object (e.g. horse) is in the position as the corresponding image in $S_{train-random152}^{clean-h}$. The we randomly choose a threshold in $\{10/255, 20/255, 30/255, ..., 250/255\}$ to generate binary noise images. All the binary images build the thresholded probability noise set $S_{train-random152}^{probability-h}$.

For images in $S_{train-random152}^{clean-h}$, we use the same method to build noise datasets $S_{train-random152}^{real-h}$, $S_{train-random152}^{occlusion-h}$ and $S_{train-random152}^{probability-h}$.

Table 6.2: Performance of Faster-RCNN.

		Mean Absolute Error (Error on x axis, Error on y axis)			
Training Set	Eval. on	clean image	real image noise	occlusion noise	thr. prob. noise
$S^{clean-h}_{train-random152}$	train	1.25(1.50, 1.00)	8.34(8.03, 8.67)	10.00(10.74, 9.27)	5.72(5.56, 5.88)
	test	1.42(1.50, 1.34)	8.57(8.13, 9.01)	9.81(10.48, 9.13)	5.63(5.36, 5.89)
$S^{real-h}_{train-random152}$	train	0.95(0.99, 0.92)	1.46(1.51, 1.41)	6.34(6.55, 6.12)	7.01(6.80, 7.22)
	test	1.51(1.56, 1.45)	1.95(2.02, 1.87)	6.50(6.72, 6.28)	7.11(6.90, 7.32)
$S^{occlusion-h}_{train-random152}$	train	1.04(1.09, 1.00)	9.50(9.40, 9.61)	1.63(1.65, 1.61)	4.27(4.16, 4.38)
	test	1.57(1.65, 1.48)	9.55(9.48, 9.63)	2.11(2.18, 2.04)	4.35(4.24, 4.46)
$S^{probability-h}_{train-random152}$	train	1.07(1.07, 1.06)	5.51(5.47, 5.55)	5.07(5.36, 4.78)	1.18(1.17, 1.18)
	test	1.37(1.38, 1.36)	5.64(5.57, 5.71)	5.09(5.41, 4.77)	1.96(2.00, 1.91)
three noise sets	train	1.26(1.30, 1.22)	2.07(2.12, 2.03)	2.27(2.33, 2.30)	1.88(1.96, 1.80)
	test	1.73(1.82, 1.64)	2.49(2.57, 2.40)	2.64(2.72, 2.56)	2.33(2.42, 2.25)

During the training process, we use $S^{clean-h}_{train-random152}$ or noisy datasets as input, the computed coordinates (x_1, y_1, x_2, y_2) as target, to train a Faster-RCNN to predict the coordinate of bounding box.

During the evaluation process, for each image in $S^{clean-h}_{test-random152}$, $S^{real-h}_{train-random152}$, $S^{occlusion-h}_{train-random152}$ and $S^{probability-h}_{train-random152}$, apply the trained Faster-RCNN model to predict the coordinates (x_1, y_1, x_2, y_2).

We try different training sets. The results are shown in Table 6.2.

6.3 DISSM Training

6.3.1 Decoder Training

The aligned images in $S^{clean-h}_{train-256}$ are used to train the decoder. The final decoder is the same as the best one in section 5.2.8. We trained 2000 epochs, in the first 200 epochs, we use the L1 loss function and update parameters of network and latent vector every epoch. In the following 900 epochs, we use the IoULoss instead of L1Loss. In the last 900 epochs, we keep using IoULoss, but only update latent vector once every 100 epochs. The learning rate of network is initially set to 0.005 and multiplied by 0.8 every 100 epochs. The learning rate of latent vector is initially set to 0.001 and multiplied by 0.8 every 100 epochs.

6.3.2 Encoder Training

Object detection. In the object detection step, we use the encoder part of the DISSM to predict translation parameter from the padded image with size 560×560, then place the object to the center of the image based on the predicted parameters, then use the centered image with size 560×560 as input to the next step.

The training set is $S^{clean-h}_{train-ori560}$. We use Vit-Base as the encoder, the input of the encoder has two layers, one is the corresponding image in $S^{clean-h}_{train-ori560}$ with size 560×560, the other is the signed distance map with size 560×560 generated by decoder from current predicted latent vector (initialized as the mean latent vector in the training of decoder). The output of the encoder is the correction value of the transformation parameters $\mathbf{t}_{k-corr} \in \mathbb{R}^2$. We use the IoUloss function for training. The learning rate of network is initially set to 0.0001.

Based on the predicted parameters $\mathbf{t}_{k-corr}$, we centered all the images in $S^{clean-h}_{train-ori560}$ and build a new training set $S^{clean-h}_{train-ori560-center}$. Then we use the new training set as input to feed another encoder.

Parameter prediction. The second encoder is Vit-Base. The input of the encoder has two layers, one is the corresponding image in $S^{clean-h}_{train-ori560-center}$ with size 560×560, the other is the signed distance map with size 560×560 generated by decoder from current predicted latent vector (initialized as the mean latent vector in the training of decoder). The output of encoder has two parts, one is the correction value of the latent vector $\mathbf{z}_{k-corr} \in \mathbb{R}^{64}$, the other is the correction value of the transformation parameters $\mathbf{t}_{k-corr} \in \mathbb{R}^2$, $s_{k-corr} \in \mathbb{R}$.

In order to scale the output values of the encoder to the same order of magnitude, we compute the standard deviation of all 159 latent vectors obtained during the training process $\sigma_{\mathbf{z}} \in \mathbb{R}^{64}$. During the alignment process from $S^{clean-h}_{train-original}$ to $S^{clean-h}_{train-256}$, we obtained the translation and scale parameters, and we compute the mean and standard deviation of these parameters as $\mu_{trans} \in \mathbb{R}^2, \sigma_{trans} \in \mathbb{R}, \mu_{scale} \in \mathbb{R}^2, \sigma_{scale} \in \mathbb{R}$. Then we update the latent vector and transformation parameters as:

$$\begin{aligned} \mathbf{z}_{k-new} &= \mathbf{z}_{k-cur} + \mathbf{z}_{k-corr} * \sigma_{\mathbf{z}}, \\ \mathbf{t}_{k-new} &= \mathbf{t}_{k-corr} * \sigma_{trans}, \\ s_{k-new} &= s_{k-corr} * \sigma_{scale}. \end{aligned} \tag{6.2}$$

We use the IoUloss function plus a penalty for bad scale prediction. The learning rate of network is initially set to 0.00011 and multiplied by 0.3 when the loss value reaches a plateau.

We compute the IoU of reconstructed images and original images. The results are displayed in Table 6.3.

Table 6.3: Reconstruction performance(IoU) of DISSM.

network architecture	train(IoU)	test(IoU)
DISSM	0.91	0.706

We compute the IoU of Denoised images from $S^{all-h}_{test-original}$. The results are displayed in Table 6.4.

Table 6.4: Denoising performance(IoU) of DISSM.

network architecture	$S^{real-h}_{train-original}$	$S^{occlusion-h}_{train-original}$	$S^{detection-h}_{train-original}$	$S^{probability-h}_{train-original}$
DISSM	0.461	0.532	0.644	0.590

6.4 U-Net and MAE Training

6.4.1 One Step Training 1

We use the training sets $S^{real-h}_{train-original}$, $S^{occlusion-h}_{train-original}$, $S^{probability-h}_{train-original}$ to train the U-Net. For each image of the training sets, eight images are generated with different sizes, each 0.841 the size of the previous one, and random 128×128 patches are cropped from these generated images as input for the U-Net. Random rotation (degree=15) is also added to these inputs. The architecture of U-Net is the same as in Section 5.2.5.

We compute the IoU of the denoised images from $S^{all-h}_{test-original}$ and original images. The results are displayed in Table 6.5.

Table 6.5: Denoising performance(IoU) of U-Net.

network architecture	$S^{real-h}_{train-original}$	$S^{occlusion-h}_{train-original}$	$S^{detection-h}_{train-original}$	$S^{probability-h}_{train-original}$
U-Net	0.919	0.869	0.832	0.864

6.4.2 One Step Training 2

We use the training sets $S^{real-h}_{train-256}$, $S^{occlusion-h}_{train-256}$, $S^{probability-h}_{train-256}$ to train the U-Net and MAE. The architecture of U-Net and MAE is the same as section 5.2.5 and 5.2.7.

We compute the IoU of Denoised images from $S^{all-h}_{test-original}$ and original images. The results are displayed in Table 6.6.

Table 6.6: Denoising performance (IoU) of U-Net and MAE.

network architecture	$S^{real-h}_{train-original}$	$S^{occlusion-h}_{train-original}$	$S^{detection-h}_{train-original}$	$S^{probability-h}_{train-original}$
U-Net	0.918	0.871	0.843	0.868
MAE	0.921	0.896	0.863	0.888

6.4.3 Two Step Training

For each image in $S^{real-h}_{train-256}$, $S^{occlusion-h}_{train-256}$, $S^{probability-h}_{train-256}$, we apply a pre-trained Faster-RCNN to predict the coordinates (x_1, y_1, x_2, y_2) of the object, adjust the coordinate to make it a square, then crop a square patch. The square patches are resized to 128×128 size and are used as input to train the U-Net and MAE.

We compute the IoU of denoised images from $S^{all-h}_{test-original}$ and original images. The results are displayed in Table 6.7.

Table 6.7: Denoising performance (IoU) of U-Net and MAE.

network architecture	$S^{real-h}_{train-original}$	$S^{occlusion-h}_{train-original}$	$S^{detection-h}_{train-original}$	$S^{probability-h}_{train-original}$
U-Net	0.901	0.871	0.863	0.866
MAE	0.883	0.861	0.861	0.866

6.5 PCA-UNet and PCA

6.5.1 Decoder (PCA Computation)

We will evaluate a one PCA model on 128×128 pixel aligned shapes and PCA models on grids of patches on 256×256 pixel aligned shapes. All PCA models have 64 principal coefficients (PCs).

Table 6.8: Shape reconstruction capability (IoU) on S_{test}^{m-all} of PCA models trained with different numbers of perturbations per image: average IoU (std) obtained over 5 runs.

Patch size	# PCA	Img. size	Number of perturbations per training shape 1	10	100	1000
Weizmann Horse dataset						
128	1	128	0.746(.029)	0.860(.002)	0.872(.001)	0.874(.000)
	2×2	256	0.864(.013)	0.930(.001)	0.938(.000)	0.939(.000)
64	4×4	256	0.902(.008)	0.964(.000)	0.970(.000)	0.971(.000)
32	8×8	256	0.935(.019)	0.982(.000)	0.987(.000)	0.988(.000)
	1	256	0.984(.000)	0.987(.000)	0.987(.000)	0.987(.000)
CUB 200 Birds dataset						
128	1	128	0.813(.023)	0.886(.001)	0.896(.000)	0.897(.000)
	2×2	256	0.882(.014)	0.942(.001)	0.948(.000)	0.949(.000)
64	4×4	256	0.924(.004)	0.962(.000)	0.970(.000)	0.971(.000)
32	8×8	256	0.939(.011)	0.982(.000)	0.988(.000)	0.989(.000)
	1	256	0.985(.000)	0.987(.000)	0.987(.000)	0.987(.000)
Horses & Birds combined dataset						
32	1	256	0.986(.000)	0.987(.000)	0.987(.000)	0.987(.000)

We did not train a single PCA model on 256×256 because it involves working with a 65536×65536 matrix, which takes 18GB of memory and is computationally prohibitive to run SVD on it.

The shape reconstruction capability is measured by the IOU of reconstructing the input shape from the PCA coefficients. The PCAs were trained using online PCA and different numbers of data augmentation perturbations per image. The data augmentation perturbations were: random resizing the 256×256 input in the interval $[241, 271]$, random rotation up to ± 15 degrees, and random cropping a 256×256 image with padding up to 15 pixels.

The reconstruction results on the test set (which was not used for training the PCAs) are shown in Table 6.8. From Table 6.8 one can see that a grid of 8×8 PCAs with patch size 32×32 and 64 PCs can do a very good job reconstructing the shapes.

Moreover, a single PCA trained on the 32×32 pixel patches can do as good a job as a grid of PCAs.

Out of curiosity, we also trained a single PCA on the combined Horse and Birds train sets and show the reconstruction results on the combined test sets in Table 6.8. One can see that the reconstruction (with at least 10 perturbations per image) is as good as the one trained on a single

Table 6.9: Shape reconstruction IoUs on the PascalVOC validation set of PCA models trained on either the horses & birds dataset or the PascalVOC train set: average IoU and std obtained over 5 runs.

	Trained on			Trained on	
Object	Horses&Birds	VOC	Obj	Horses&Birds	VOC
aeroplane	0.954(.000)	0.955(.000)	bus	0.995(.000)	0.995(.000)
bicycle	0.813(.000)	0.811(.000)	car	0.977(.000)	0.976(.000)
bottle	0.982(.000)	0.981(.000)	chair	0.938(.000)	0.939(.000)
diningtable	0.993(.000)	0.993(.000)	dog	0.990(.000)	0.990(.000)
horse	0.980(.000)	0.980(.000)	train	0.993(.000)	0.993(.000)
motorbike	0.977(.000)	0.977(.000)	cow	0.986(.000)	0.985(.000)
person	0.974(.000)	0.974(.000)	sofa	0.996(.000)	0.995(.000)
pottedplant	0.876(.000)	0.875(.000)	cat	0.994(.000)	0.994(.000)
sheep	0.978(.000)	0.978(.000)	boat	0.959(.000)	0.961(.000)
tvmonitor	0.994(.000)	0.995(.000)	bird	0.968(.000)	0.968(.000)

dataset (e.g. Horse). This might mean that the trained PCA on 32×32 patches is capable of reconstructing arbitrary shapes.

To test this hypothesis, we used the shape model trained on the combined Horses & Birds train sets to reconstruct the shapes (segmentations) of the the PascalVOC validation set. The results are shown in Table 6.9. Also shown are results of equivalent PCA models trained on the segmentations of the PascalVOC train set. The results are very similar, sometimes differing by 0.001. From Table 6.9 one could see that most of the shapes can be accurately reconstructed by the PCA model, except for the bicycles and potted plants.

To illustrate the effect of the model trained on the combined Horses & Birds training sets, we prepare two 32×32 image patches: one is split horizontally with a white top half and black bottom half (patch 1), and the other is split vertically with a white left half and black right half (patch 2). We then compute the two PCA coefficient vectors for each patch using the model trained on the combined Horses & Birds data.

Next, we interpolate between these two coefficient vectors to generate eight intermediate vectors. By reconstructing the image patches from these ten vectors, we observe a gradual transformation from one reconstructed shape to the other in the Figure 6.3.

To understand why the PCA model trained on the combined Horses & Birds training sets struggles to reconstruct bicycle shapes, we trained a new PCA model using only the bicycle shapes

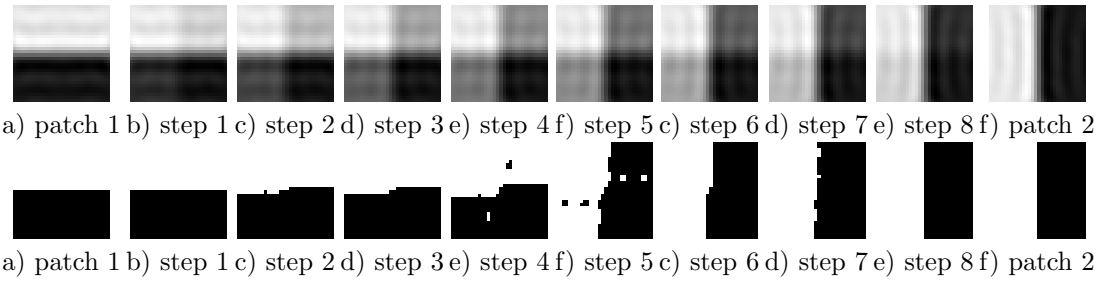


Figure 6.3: Transformation path between two reconstructed patches, the first row shows the original reconstructed images, the second row shows the shape results after applying a threshold (0.5)

from the PascalVOC training set. Testing this model on bicycle shapes from the PascalVOC validation set yielded a reconstruction IoU of 0.8385, which is still lower than that of other objects in the PascalVOC dataset.

To visualize the differences between the two models, we prepared two 32×32 image patches: one containing a horizontal white ridge (3 pixels thick) centered on a black background (patch 3) and the other containing a vertical white ridge on the same black background (patch 4). We computed the coefficient vectors for these patches using both the Horses & Birds PCA model and the bicycle-only PCA model and interpolated between the vectors for each model. The transformation paths are shown in Figure 6.4.

6.5.2 One Step Training

We use the training sets $S^{real-h}_{train-384}$, $S^{occlusion-h}_{train-384}$, $S^{probability-h}_{train-384}$ to train the PCA-UNet. Data augmentation for the the training set includes random resizing to (384 - 15, 384 + 15), random rotation (degree=15) and random cropping (output size is 384, padding size on each border of the image is 15, pads with the last value at the edge of the image). The backbone of the encoder is ResNet50, the number of PCA components is 64. The PCA components are pre-trained on the combined horse and bird train datasets.

We compute the IoU of the denoised images from $S^{all-h}_{test-original}$ and original images. The results are displayed in Table 6.5.

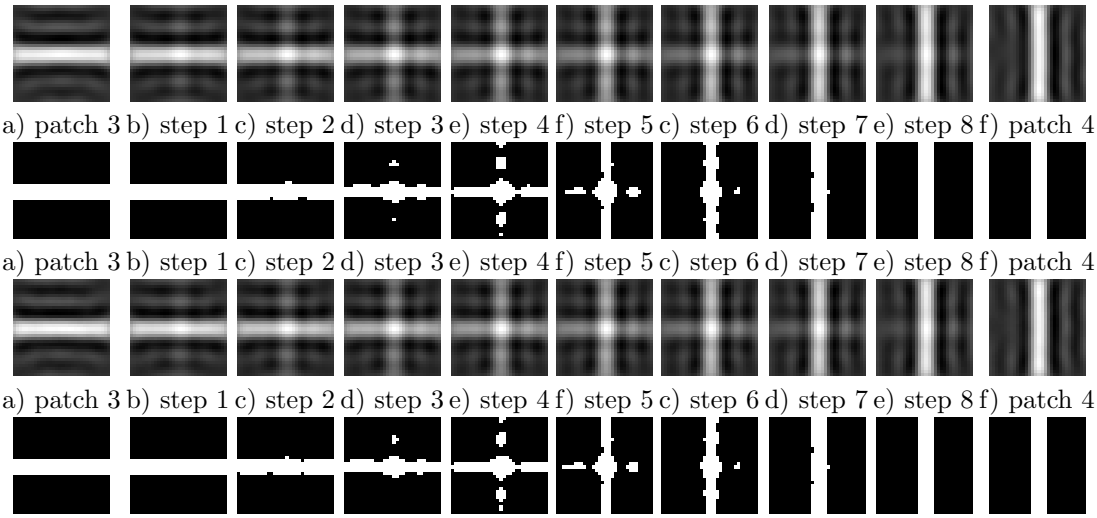


Figure 6.4: Transformation path comparison between two reconstructed patches, the first two rows are the original reconstructed images (first row) and the shape results (second row) from combined Horses & Birds training sets, the last two rows are the original reconstructed images (third row) and the shape results(fourth row) from the bicycle-only PCA model

Table 6.10: Denoising performance(IoU) of PCA-UNet.

network architecture	$S^{real-h}_{train-original}$	$S^{occlusion-h}_{train-original}$	$S^{detection-h}_{train-original}$	$S^{probability-h}_{train-original}$
PCA-UNet	0.945	0.910	0.867	0.900
PCA-UNet-ATT	0.946	0.914	0.870	0.904

6.5.3 Two Step Training

For each image in $S^{real-h}_{train-256}$, $S^{occlusion-h}_{train-256}$, $S^{probability-h}_{train-256}$, we apply a pre-trained Faster-RCNN to predict the coordinates (x_1, y_1, x_2, y_2) of the object, and adjust the coordinate to make it a square, then crop the square patch. The square patches are resized to 128×128 size and used as input to train the PCA-UNet.

We compute the IoU of the denoised images from $S^{all-h}_{test-original}$ and original images. The results are displayed in Table 6.11.

Table 6.11: Denoising performance (IoU) of PCA-UNet with Faster-RCNN.

network architecture	$S^{real-h}_{train-original}$	$S^{occlusion-h}_{train-original}$	$S^{detection-h}_{train-original}$	$S^{probability-h}_{train-original}$
FasterRCNN+PCA-UNet	0.944	0.907	0.872	0.899
FasterRCNN+PCA-UNet-ATT	0.943	0.906	0.977	0.902

6.5.4 PCA

In our ablation study, we examine the impact of using Principal Component Analysis (PCA) alone for shape denoising, omitting the neural network encoder. Specifically, for a given noisy shape patch, denoted as $\mathbf{S}_n$, we compute its PCA coefficients $\mathbf{c}$ directly through:

$$\mathbf{c} = \mathbf{P}^T \mathbf{S}_n \tag{6.3}$$

where $\mathbf{P}$ is the PCA components matrix.

Same as PCA-UNet, we also tried one step and two step methods, denoted as 'PCA-One' and 'PCA-Two' in our tables and figures.

6.6 Shape Denoising in the Wild Evaluation

In this section, we assess the robustness of the proposed shape denoising methods on real-world noise scenarios. These evaluations are designed to simulate realistic challenges that occur in natural environments. By testing these models on a variety of noise conditions, including real image noise, occlusion noise, detection image noise, and thresholded probability noise, we can determine their performance in the wild.

6.6.1 Real Image Noise

An example of denoising results obtained by the methods evaluated on a shape corrupted by real image noise is shown in Fig. 6.5. The comparison of all methods is displayed in Table 6.12 and Fig. 6.6 (left).

The Horse Dataset. The PCA-UNet variants dominate in terms of performance. PCA-UNet-ATT stands out with the highest average IoU of 0.946, closely followed by PCA-UNet, PCA-UNet-Two and PCA-UNet-ATT-Two with an average IoU of 0.945, 0.944 and 0.943. All these

PCA-UNet variants show exceptional results, particularly when the noise level is high, indicating their robustness in handling real-world noise.

MAE-One, UNet-One and UNet-One-256 also perform strongly, with average IoUs of 0.921, 0.919, and 0.918, respectively, demonstrating their efficiency in recovering shapes even under significant noise.

UNet-Two and MAE-Two provide intermediate results, with average IoUs of 0.901 and 0.883, respectively.

The Bird Dataset. The PCA-UNet outperforms all other methods across all 10 categories, with an impressive average IoU of 0.929. Additionally, its variants also surpass other models in performance.

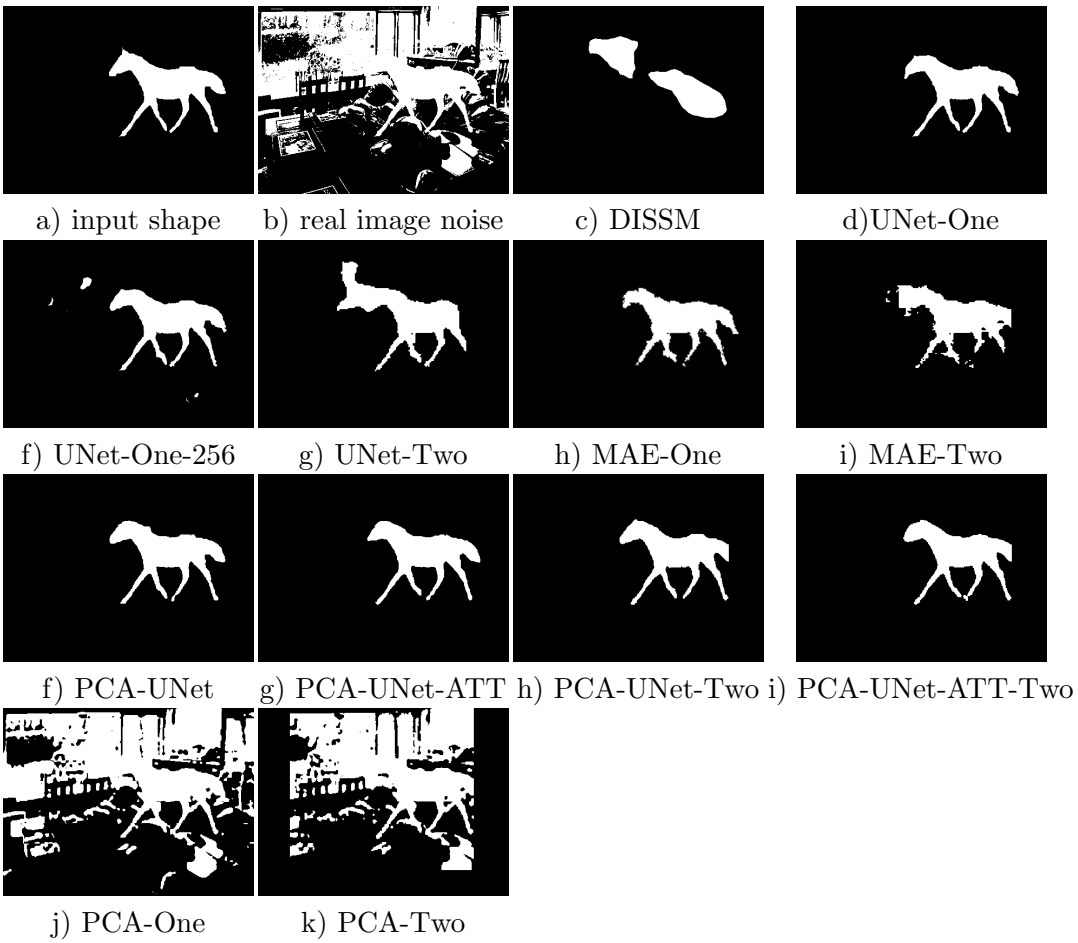


Figure 6.5: Example of results of different methods on a shape perturbed by real image noise.

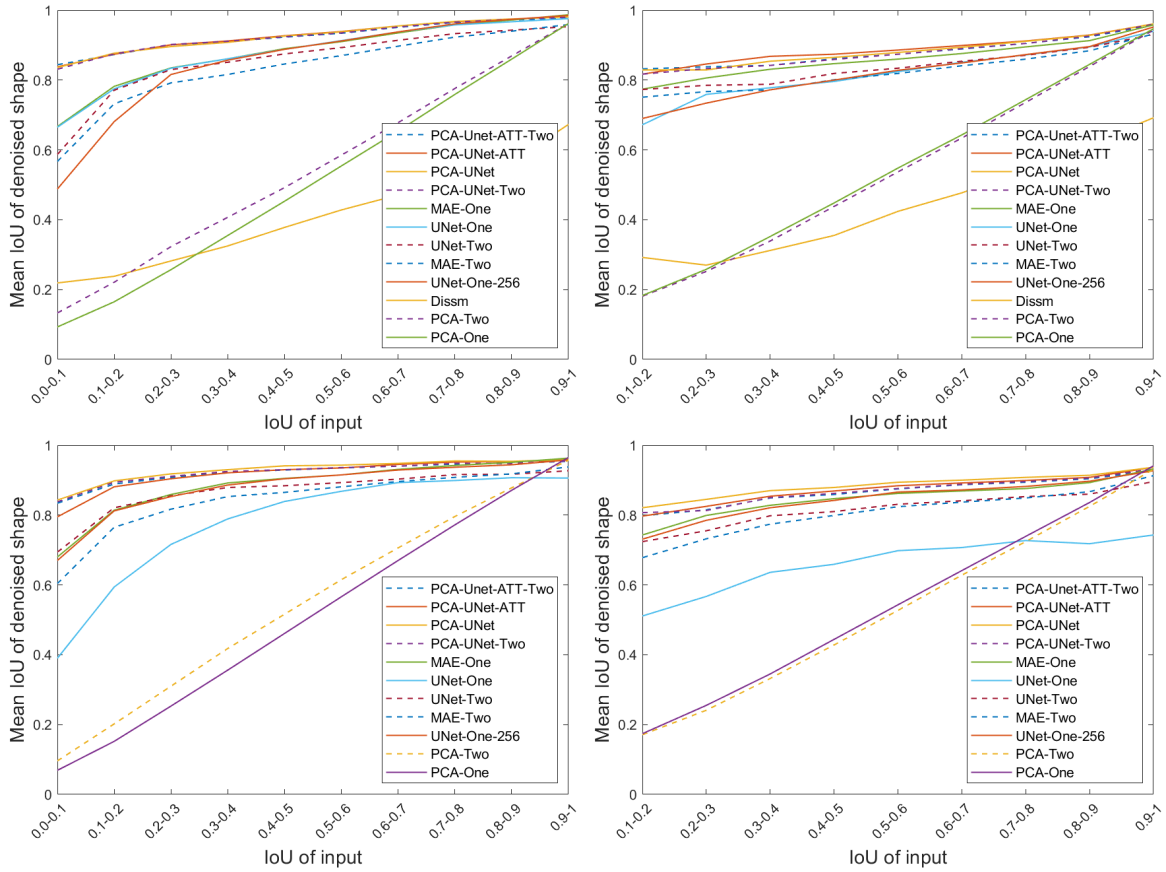


Figure 6.6: Performance on real image noise $S^{real-h}_{test-original}$ (top left), $S^{real-b}_{test-original}$ (bottom left) and occlusion noise $S^{occlusion-h}_{test-original}$ (top right), $S^{occlusion-b}_{test-original}$ (bottom right).

6.6.2 Occlusion Noise

An example of denoising results obtained by the methods evaluated on a shape corrupted by occlusion noise is shown in Fig. 6.7. The comparison of all methods is displayed in Table 6.13 and Fig. 6.6 (right).

The Horse Dataset. The PCA-UNet variants dominate in terms of performance. PCA-UNet-ATT stands out with the highest average IoU of 0.924, closely followed by PCA-UNet, PCA-UNet-Two and PCA-UNet-ATT-Two, with average IoUs of 0.921, 0.918 and 0.917, respectively. All these PCA-UNet variants show exceptional results, particularly at higher noise levels, reflecting their robustness in managing occlusion noise.

MAE-One delivers strong performance with an average IoU of 0.909, indicating its effectiveness in recovering shapes even under significant occlusion.

UNet-One-256, UNet-One and UNet-Two also perform strongly, with average IoUs of 0.889, 0.885 and 0.885, respectively. These models demonstrate consistency across noise levels, making them effective options for managing occlusion noise.

MAE-Two provide intermediate results with average IoU of 0.877.

The Bird Dataset. The PCA-UNet is the best method across all 10 categories, with an impressive average IoU of 0.899. Additionally, its variants also surpass other models in performance.

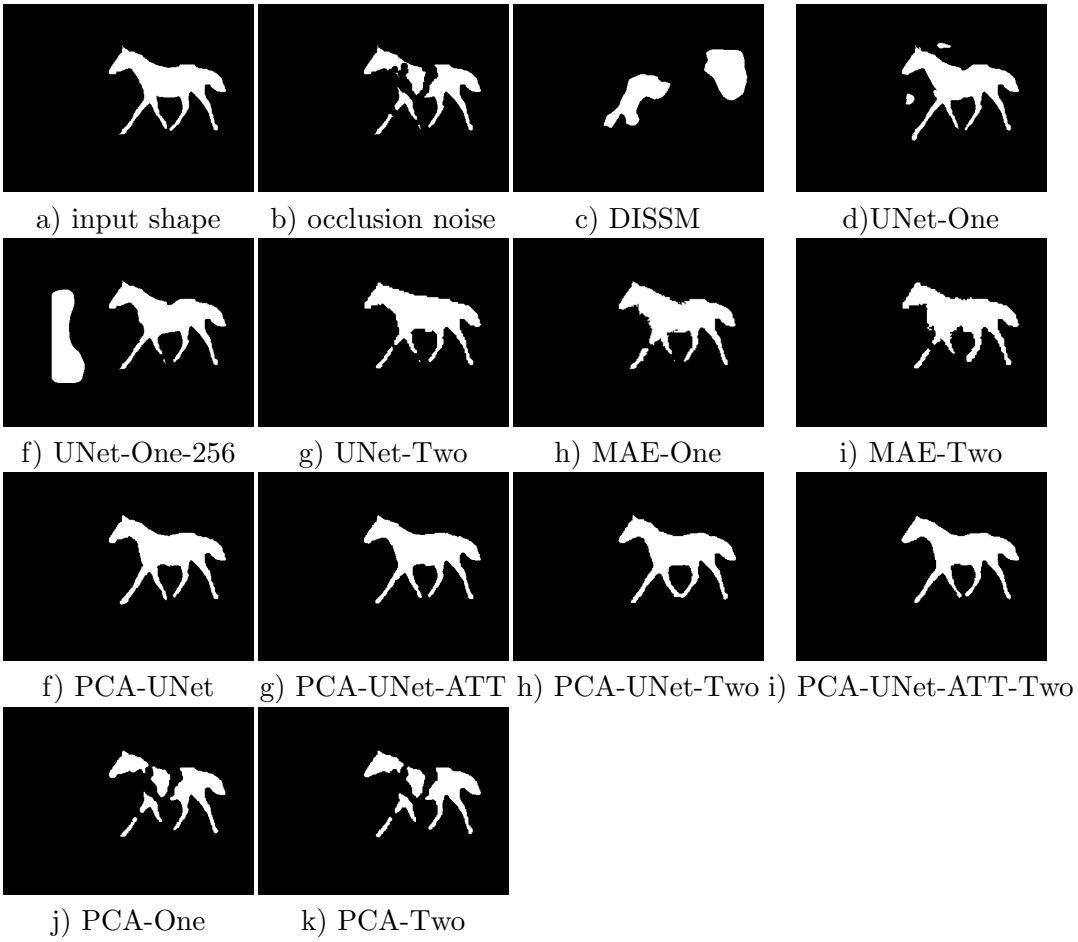


Figure 6.7: Example of results of different methods on a shape perturbed by occlusion noise.

6.6.3 Detection Image Noise

An example of denoising results obtained by the methods evaluated on a shape corrupted by detection image noise is shown in Fig. 6.8. The comparison of all methods is displayed in Table 6.14 and Fig. 6.9.

The Horse Dataset. The PCA-UNet variants perform exceptionally well in this task. PCA-UNet-ATT-Two achieves the highest average IoU of 0.877, closely followed by PCA-UNet-Two, PCA-UNet-ATT and PCA-UNet, with average IoUs of 0.872, 0.870 and 0.869, respectively.

UNet-Two, MAE-One and MAE-Two also perform strongly, with average IoUs of 0.863, 0.863 and 0.861, respectively.

UNet-One-256 and UNet-One provide intermediate results, with average IoUs of 0.843 and 0.832, respectively.

The Bird Dataset. The PCA-UNet-ATT-Two and PCA-UNet-Two are the top performers, with average IoUs of 0.726 and 0.723. U-Net-Two follows with an average IoU 0.712.

6.6.4 Thresholded Probability Noise

An example of denoising results obtained by the methods evaluated on a shape corrupted by thresholded probability noise is shown in Fig. 6.10. The comparison of all methods is displayed in Table 6.15 and Fig. 6.9.

The Horse Dataset. The PCA-UNet variants dominate in terms of performance. PCA-UNet-ATT stands out with the highest average IoU of 0.904, closely followed by PCA-UNet-ATT-Two, PCA-UNet-Two and PCA-UNet, with average IoUs of 0.902, 0.900 and 0.900, respectively. All these PCA-UNet variants show exceptional results, particularly at higher noise levels, reflecting their robustness in managing occlusion noise.

MAE-One also shows solid results, with an average IoU of 0.889. It excels in the higher IoU range, reaching 0.942, which makes it a competitive method for handling thresholded probability noise.

UNet-One-256, UNet-Two, MAE-Two and UNet-One has similar performance, with average IoUs of 0.868, 0.866, 0.866 and 0.864, respectively.

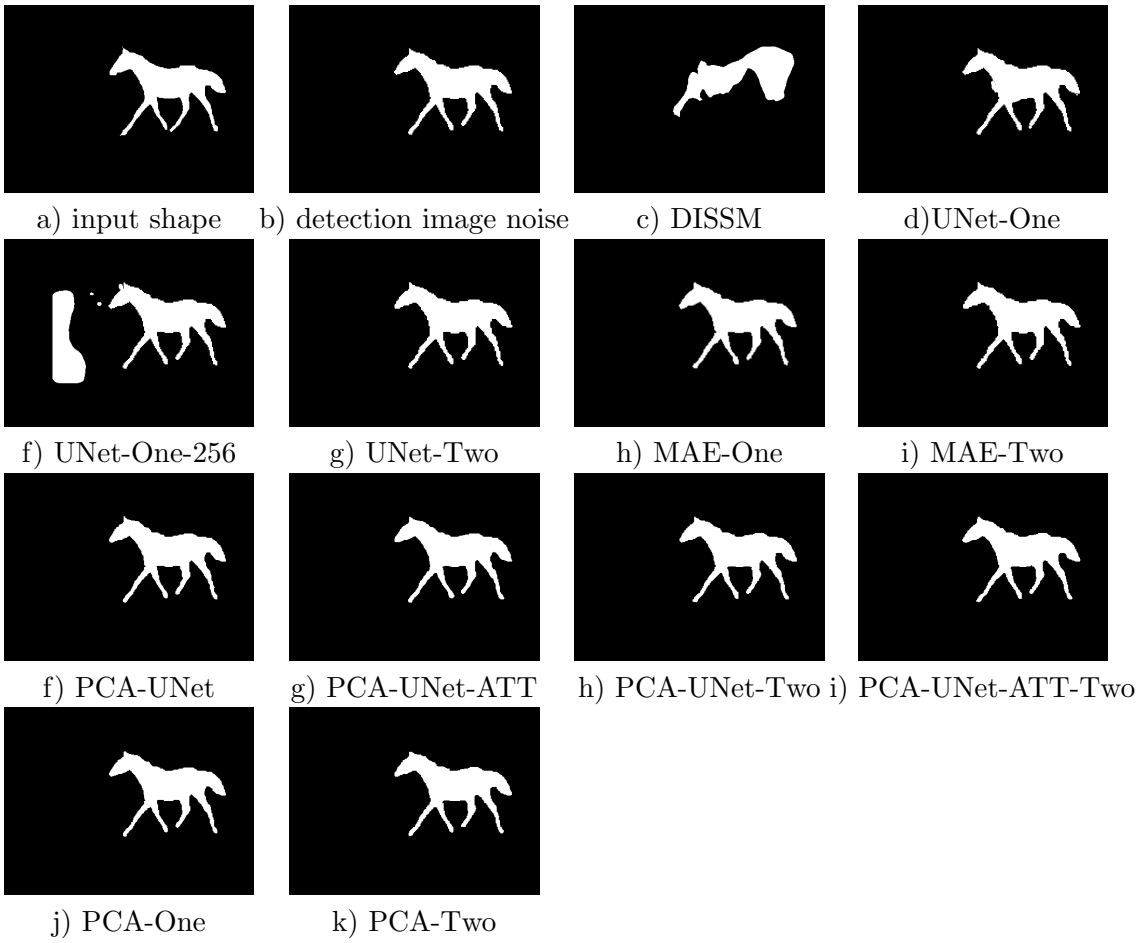


Figure 6.8: Example of results of different methods on a shape perturbed by detection image noise.

The Bird Dataset. The PCA-UNet variants dominate in terms of performance, with average IoUs of 0.777 (PCA-UNet), 0.773 (PCA-UNet-Two), 0.773 (PCA-UNet-ATT-Two) and 0.759 (PCA-UNet-ATT). The next method is MAE with an average IoU of 0.715.

6.6.5 Overall Comparison of Shape Denoising Across Noise Types

In this section, we present a comprehensive comparison of the denoising methods across all four noise types: real image noise, occlusion noise, detection image noise, and thresholded probability noise. The table below 6.16 shows the average Intersection over Union (IoU) performance of each method for the respective noise types, providing a clear comparison of these methods.

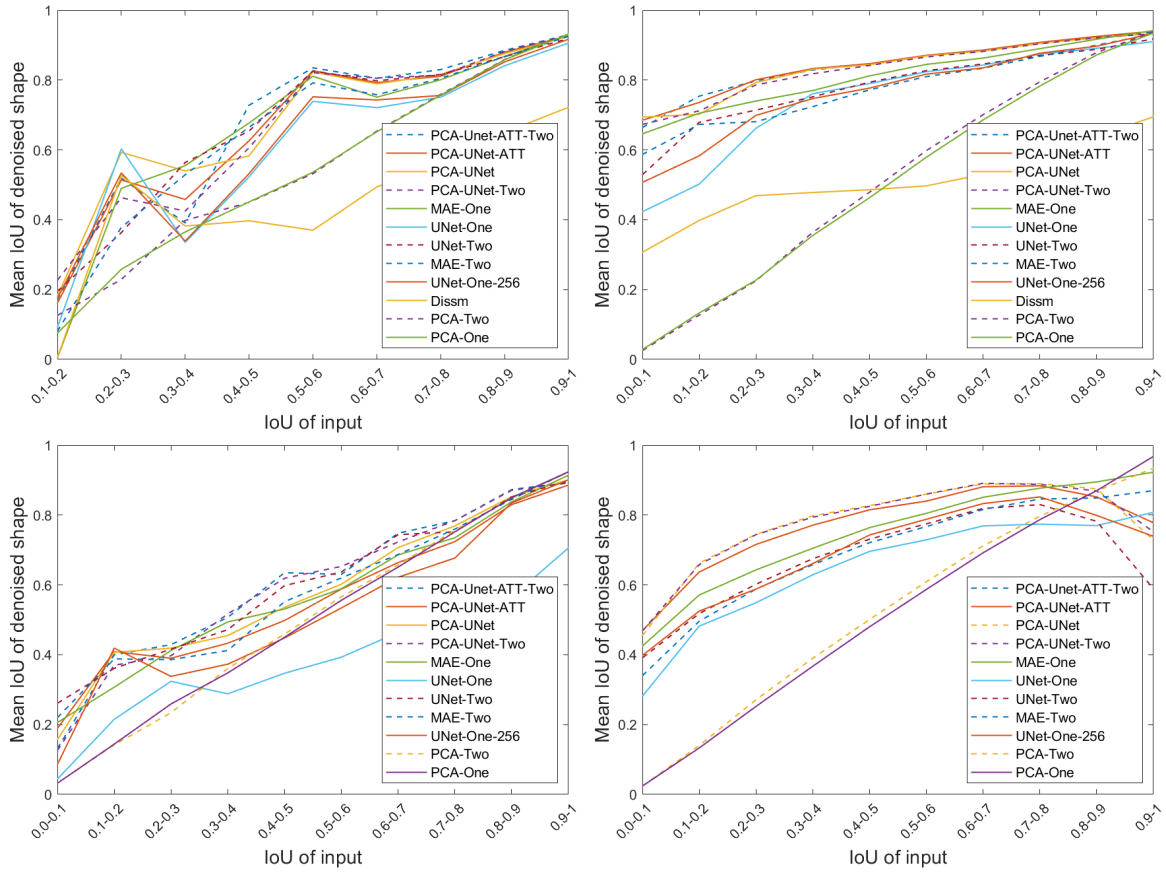


Figure 6.9: Performance on detection image noise $S^{detection-h}_{test-original}$ (top left), $S^{detection-b}_{test-original}$ (bottom left) and thresholded probability noise $S^{probability-h}_{test-original}$ (top right), $S^{probability-b}_{test-original}$ (bottom right).

The Figure 6.11 provides a summary of how each method performs in various noisy environments, giving insights into the strengths of PCA-UNet variants and the overall reliability of each approach.

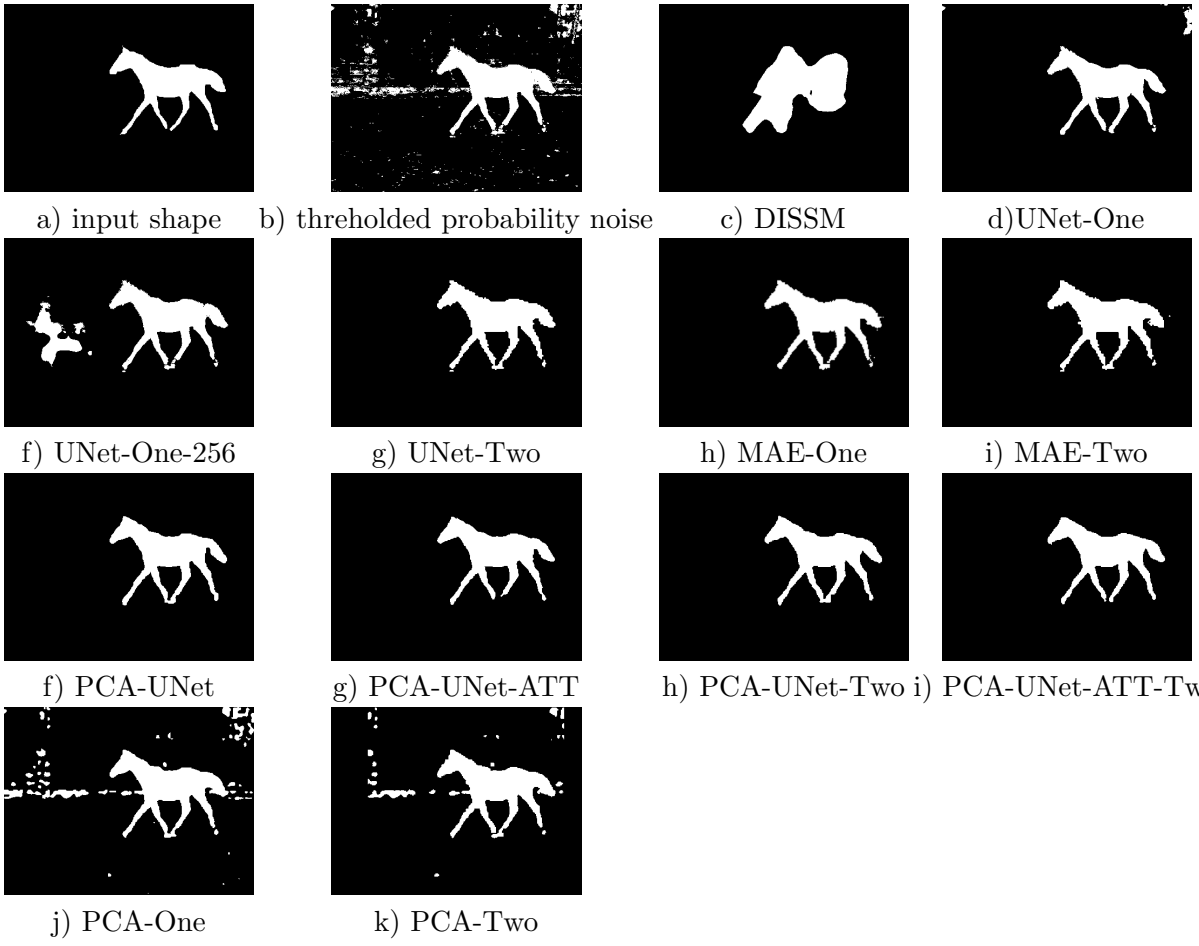


Figure 6.10: Example of results of different methods on a shape perturbed by thresholded probability noise.

Table 6.12: Comparison of methods against real image noise.

Method	IoU(begin)										avg
	0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	
	$S^{real-h}_{test-original}$										
DISSM	0.219	0.238	0.282	0.325	0.378	0.428	0.471	0.514	0.565	0.673	0.461
UNet-One	0.665	0.775	0.834	0.861	0.888	0.913	0.938	0.958	0.967	0.976	0.919
UNet-One-256	0.488	0.681	0.816	0.857	0.888	0.912	0.938	0.961	0.974	0.985	0.918
UNet-Two	0.587	0.771	0.83	0.851	0.875	0.893	0.914	0.933	0.943	0.953	0.901
MAE-One	0.667	0.782	0.835	0.861	0.89	0.91	0.935	0.958	0.972	**0.987**	0.921
MAE-Two	0.567	0.732	0.792	0.816	0.846	0.87	0.897	0.923	0.94	0.959	0.883
PCA-UNet	0.836	**0.877**	0.896	0.908	0.926	0.938	**0.955**	**0.968**	**0.975**	0.981	0.945
PCA-UNet-ATT	0.838	0.874	0.901	**0.912**	**0.927**	**0.939**	**0.955**	0.967	**0.975**	0.981	**0.946**
PCA-UNet-Two	0.831	0.875	**0.902**	0.911	0.924	0.935	0.952	0.966	0.974	0.98	0.944
PCA-UNet-ATT-Two	**0.844**	0.873	0.901	0.91	0.923	0.934	0.951	0.965	0.972	0.978	0.943
PCA-One	0.093	0.165	0.257	0.355	0.453	0.554	0.654	0.759	0.860	0.962	0.620
PCA-Two	0.133	0.221	0.323	0.407	0.492	0.585	0.678	0.776	0.869	0.962	0.649
	$S^{real-b}_{test-original}$										
UNet-One	0.390	0.594	0.716	0.789	0.839	0.868	0.893	0.899	0.907	0.906	0.780
UNet-One-256	0.670	0.812	0.853	0.886	0.904	0.915	0.929	0.937	0.944	0.958	0.881
UNet-Two	0.694	0.821	0.855	0.879	0.884	0.893	0.903	0.916	0.917	0.927	0.869
MAE-One	0.680	0.814	0.859	0.892	0.905	0.915	0.931	0.942	0.951	0.963	0.886
MAE-Two	0.604	0.765	0.817	0.853	0.865	0.881	0.896	0.908	0.918	0.938	0.845
PCA-UNet	**0.843**	**0.898**	**0.918**	**0.930**	**0.941**	**0.943**	**0.948**	**0.955**	**0.954**	0.959	**0.929**
PCA-UNet-ATT	0.795	0.881	0.904	0.921	0.930	0.935	0.945	0.952	0.953	0.958	0.918
PCA-UNet-Two	0.838	0.894	0.911	0.925	0.930	0.936	0.940	0.948	0.948	0.955	0.923
PCA-UNet-ATT-Two	0.834	0.889	0.908	0.922	0.929	0.935	0.940	0.946	0.948	0.954	0.920
PCA-One	0.069	0.152	0.253	0.356	0.461	0.566	0.670	0.772	0.871	**0.965**	0.511
PCA-Two	0.096	0.202	0.310	0.418	0.516	0.615	0.706	0.796	0.878	0.957	0.547

Table 6.13: Comparison of methods against occlusion noise.

Method	IoU(begin)									avg
	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	
	$S^{occlusion-h}_{test-original}$									
DISSM	0.292	0.27	0.312	0.355	0.424	0.477	0.543	0.617	0.692	0.570
UNet-One	0.672	0.759	0.778	0.796	0.824	0.851	0.872	0.895	0.94	0.885
UNet-One-256	0.69	0.734	0.772	0.8	0.828	0.849	0.873	0.896	0.952	0.889
UNet-Two	0.773	0.785	0.788	0.819	0.834	0.854	0.871	0.894	0.931	0.885
MAE-One	0.774	0.806	0.831	0.847	0.86	0.876	0.895	0.913	0.958	0.909
MAE-Two	0.751	0.767	0.772	0.801	0.819	0.841	0.86	0.884	0.932	0.877
PCA-UNet	0.829	0.829	0.854	0.865	0.879	0.894	0.911	0.928	**0.961**	0.921
PCA-UNet-ATT	0.817	**0.846**	**0.868**	**0.874**	**0.886**	**0.899**	**0.912**	**0.929**	**0.961**	**0.924**
PCA-UNet-Two	0.817	0.833	0.842	0.859	0.874	0.89	0.905	0.926	0.96	0.918
PCA-UNet-ATT-Two	**0.832**	0.838	0.842	0.861	0.874	0.889	0.905	0.924	0.958	0.917
PCA-One	0.183	0.258	0.352	0.448	0.548	0.643	0.744	0.845	0.947	0.770
PCA-Two	0.181	0.252	0.339	0.438	0.537	0.634	0.736	0.839	0.943	0.763
	$S^{occlusion-b}_{test-original}$									
U-Net-One	0.511	0.567	0.636	0.659	0.698	0.707	0.727	0.718	0.743	0.696
U-Net-One-256	0.731	0.785	0.821	0.842	0.866	0.872	0.882	0.897	0.927	0.871
U-Net-Two	0.724	0.755	0.798	0.810	0.831	0.841	0.853	0.860	0.896	0.840
MAE-One	0.743	0.799	0.828	0.847	0.862	0.869	0.876	0.893	0.932	0.871
MAE-Two	0.678	0.732	0.774	0.799	0.824	0.837	0.849	0.867	0.913	0.836
PCA-UNet	**0.821**	**0.845**	**0.870**	**0.879**	**0.894**	**0.900**	**0.908**	**0.914**	0.937	**0.899**
PCA-UNet-ATT	0.797	0.825	0.854	0.869	0.884	0.892	0.899	0.908	0.937	0.891
PCA-UNet-Two	0.807	0.813	0.850	0.862	0.876	0.888	0.896	0.906	0.932	0.886
PCA-UNet-ATT-Two	0.798	0.815	0.849	0.859	0.875	0.887	0.894	0.904	0.928	0.884
PCA-One	0.174	0.255	0.345	0.444	0.543	0.641	0.739	0.836	**0.941**	0.639
PCA-Two	0.171	0.241	0.332	0.428	0.527	0.628	0.724	0.825	0.931	0.626

Table 6.14: Comparison of methods against detection image noise.

Method	IoU(begin)									avg
	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	
	$S^{detection-h}_{test-original}$									
DISSM	0.005	0.527	0.382	0.397	0.37	0.494	0.558	0.65	0.722	0.644
UNet-One	0.092	0.603	0.335	0.523	0.739	0.721	0.751	0.841	0.906	0.832
UNet-One-256	0.175	0.534	0.339	0.532	0.752	0.743	0.756	0.852	0.917	0.843
UNet-Two	0.195	0.363	**0.564**	0.654	0.826	0.798	0.816	0.867	0.919	0.863
MAE-One	0.004	0.49	0.555	**0.677**	0.811	0.751	0.801	0.868	0.928	0.863
MAE-Two	0.081	0.378	0.529	0.665	0.792	0.758	0.805	0.867	0.925	0.861
PCA-UNet	0.181	**0.593**	0.54	0.583	0.823	0.789	0.816	0.876	0.925	0.869
PCA-UNet-ATT	0.161	0.514	0.458	0.626	0.827	0.793	0.811	0.88	0.927	0.870
PCA-UNet-Two	**0.227**	0.463	0.425	0.606	0.821	**0.806**	0.814	0.882	0.928	0.872
PCA-UNet-ATT-Two	0.171	0.519	0.391	0.728	**0.835**	0.805	**0.83**	**0.885**	0.93	**0.877**
PCA-One	0.075	0.258	0.365	0.451	0.536	0.652	0.757	0.858	**0.932**	0.840
PCA-Two	0.126	0.229	0.398	0.450	0.532	0.654	0.759	0.859	**0.932**	0.842
	$S^{detection-b}_{test-original}$									
U-Net-One	0.388	0.573	0.485	0.447	0.497	0.382	0.381	0.464	0.472	0.465
U-Net-One-256	0.609	0.715	0.588	0.703	0.664	0.605	0.585	0.640	0.637	0.640
U-Net-Two	0.894	0.705	0.640	0.750	0.753	0.640	**0.693**	0.705	0.700	0.712
MAE-One	0.904	0.775	0.708	0.785	0.732	0.637	0.632	0.676	0.668	0.685
MAE-Two	0.898	0.733	**0.778**	0.681	**0.755**	0.634	0.611	0.689	0.678	0.695
PCA-UNet	**0.925**	**0.863**	0.665	0.659	0.729	0.626	0.611	0.703	0.698	0.700
PCA-UNet-ATT	0.875	0.804	0.629	0.729	0.716	0.550	0.588	0.667	0.670	0.671
PCA-UNet-Two	0.914	0.736	0.704	0.746	0.754	**0.687**	0.658	0.723	0.714	0.723
PCA-UNet-ATT-Two	0.923	0.736	0.778	**0.793**	0.749	0.631	0.690	**0.731**	**0.715**	**0.726**
PCA-One	0.759	0.764	0.668	0.643	0.706	0.622	0.562	0.657	0.638	0.655
PCA-Two	0.783	0.734	0.668	0.643	0.718	0.635	0.562	0.659	0.637	0.659

Table 6.15: Comparison of methods against thresholded probability noise.

Method	IoU(begin)										avg
	0-0.1	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	0.6-0.7	0.7-0.8	0.8-0.9	0.9-1	
	$S^{probability-h}_{test-original}$										
DISSM	0.307	0.398	0.469	0.478	0.486	0.497	0.53	0.587	0.637	0.695	0.590
UNet-One	0.423	0.502	0.662	0.761	0.79	0.824	0.843	0.874	0.891	0.91	0.864
UNet-One-256	0.507	0.583	0.699	0.747	0.777	0.817	0.835	0.877	0.897	0.934	0.868
UNet-Two	0.529	0.679	0.714	0.752	0.794	0.827	0.846	0.87	0.888	0.918	0.866
MAE-One	0.646	0.705	0.74	0.77	0.812	0.845	0.863	0.89	0.917	**0.942**	0.889
MAE-Two	0.587	0.673	0.681	0.724	0.772	0.81	0.834	0.868	0.9	0.932	0.866
PCA-UNet	**0.694**	0.702	0.793	0.831	0.844	0.869	0.883	0.905	0.921	0.933	0.900
PCA-UNet-ATT	0.685	0.736	**0.801**	**0.833**	**0.847**	**0.871**	**0.886**	**0.908**	**0.925**	0.94	**0.904**
PCA-UNet-Two	0.673	0.712	0.786	0.818	0.842	0.866	0.882	0.904	0.921	0.933	0.900
PCA-UNet-ATT-Two	0.664	**0.754**	0.798	0.829	0.846	0.87	0.885	0.904	0.923	0.937	0.902
PCA-One	0.028	0.133	0.227	0.355	0.463	0.579	0.686	0.783	0.872	0.935	0.768
PCA-Two	0.025	0.127	0.225	0.364	0.479	0.598	0.701	0.795	0.879	0.938	0.777
	$S^{probability-b}_{test-original}$										
U-Net-One	0.281	0.482	0.549	0.629	0.696	0.729	0.769	0.774	0.770	0.808	0.624
U-Net-One-256	0.398	0.525	0.588	0.661	0.743	0.788	0.833	0.852	0.799	0.739	0.681
U-Net-Two	0.391	0.518	0.602	0.675	0.731	0.775	0.819	0.830	0.782	0.589	0.675
MAE-One	0.425	0.571	0.643	0.705	0.764	0.805	0.851	0.877	**0.895**	0.923	0.715
MAE-Two	0.340	0.496	0.588	0.658	0.721	0.767	0.816	0.846	0.849	0.870	0.664
PCA-UNet	**0.493**	**0.663**	0.743	0.789	0.833	**0.863**	**0.897**	0.887	0.859	0.623	**0.777**
PCA-UNet-ATT	0.469	0.637	0.716	0.771	0.815	0.840	0.881	0.884	0.852	0.778	0.759
PCA-UNet-Two	0.468	0.658	0.744	0.794	0.825	0.860	0.891	0.888	0.869	0.752	0.773
PCA-UNet-ATT-Two	0.455	0.662	**0.745**	**0.798**	**0.827**	0.860	0.889	**0.890**	0.875	0.729	0.773
PCA-One	0.024	0.133	0.252	0.367	0.480	0.587	0.692	0.787	0.870	**0.968**	0.434
PCA-Two	0.024	0.140	0.271	0.390	0.501	0.609	0.712	0.797	0.869	0.934	0.449

Table 6.16: Shape denoising performance (IoU) on original test masks corrupted by four types of noise $S^{all-h}_{test-original}$ and $S^{all-b}_{test-original}$.

Method	$S^{real-h}_{test-original}$	$S^{detection-h}_{test-original}$	$S^{probability-h}_{test-original}$	$S^{occlusion-h}_{test-original}$
Dissm	0.461	0.644	0.590	0.570
UNet-One	0.919	0.832	0.864	0.885
UNet-One-256	0.918	0.843	0.868	0.889
UNet-Two	0.901	0.863	0.866	0.885
MAE-One	0.921	0.863	0.889	0.909
MAE-Two	0.883	0.861	0.866	0.877
PCA-UNet	0.945	0.869	0.900	0.921
PCA-UNet-ATT	**0.946**	0.870	**0.904**	**0.924**
PCA-UNet-Two	0.944	0.872	0.900	0.918
PCA-UNet-ATT-Two	0.943	**0.877**	0.902	0.917
PCA-One	0.620	0.840	0.768	0.770
PCA-Two	0.649	0.842	0.777	0.763
	$S^{real-b}_{test-original}$	$S^{detection-b}_{test-original}$	$S^{probability-b}_{test-original}$	$S^{occlusion-b}_{test-original}$
UNet-One	0.780	0.465	0.624	0.696
UNet-One-256	0.881	0.640	0.681	0.871
UNet-Two	0.869	0.712	0.675	0.840
MAE-One	0.886	0.685	0.715	0.871
MAE-Two	0.845	0.695	0.664	0.836
PCA-UNet	**0.929**	0.700	**0.777**	**0.899**
PCA-UNet-ATT	0.918	0.671	0.759	0.891
PCA-UNet-Two	0.923	0.723	0.773	0.886
PCA-UNet-ATT-Two	0.920	**0.726**	0.773	0.884
PCA-One	0.511	0.655	0.434	0.639
PCA-Two	0.547	0.659	0.449	0.626

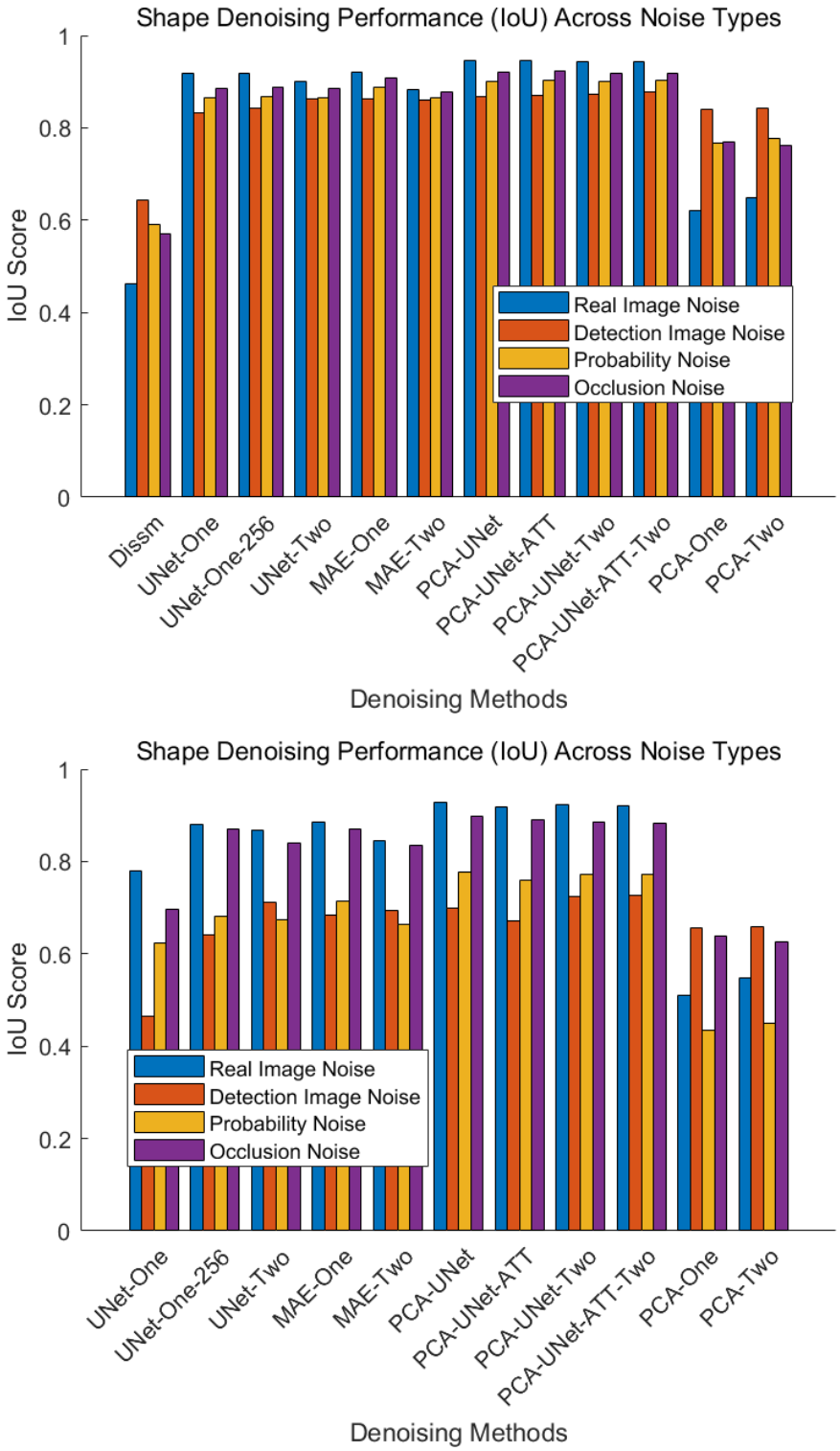


Figure 6.11: Average IoU Results of Denoising Methods across Four Noise Types on the horse dataset $S^{all-h}_{test-original}$ (top) and on the bird dataset $S^{all-b}_{test-original}$ (bottom)

CHAPTER 7

CONCLUSION

In evaluating the pros and cons of each method, the analysis is divided into two parts: **controlled** and **uncontrolled environments**.

Controlled Environment

In the controlled environment, six different types of noise were introduced: *Salt and Pepper Noise*, *Circle Noise*, *Real Image Noise*, *Occlusion Noise*, *Detection Image Noise*, and *Thresholded Probability Noise*. The findings indicate that **PCA-UNet**, **MAE**, and **U-Net** consistently outperform the other methods across most noise categories:

- **PCA-UNet** excels in *Circle Noise*, *Detection Image Noise*, and *Thresholded Probability Noise*, demonstrating its robustness in challenging noise scenarios.
- **MAE** and **U-Net** are particularly strong in handling *Salt and Pepper Noise*, *Real Image Noise*, and *Occlusion Noise*, showing their efficiency in less complex noise conditions.

DeepLabv3+ performs moderately well across all noise types, consistently ranking just below the top performers. **EBM** outperforms **CDBM** across all six noise types, especially excelling in *Real Image Noise*.

In terms of difficulty:

- *Salt and Pepper Noise* is the easiest to handle, followed by *Real Image Noise*.
- *Circle Noise* and *Occlusion Noise* are more challenging, especially at higher noise levels.
- *Detection Image Noise* and *Thresholded Probability Noise* are the most difficult to deal with.

Uncontrolled (Wild) Environment

In the wild environment, four noise types were introduced: *Real Image Noise*, *Occlusion Noise*, *Detection Image Noise*, and *Thresholded Probability Noise*. The results emphasize the dominance of the **PCA-UNet variants** across all noise types, showcasing their superior performance.

- In the **PCA-UNet** family, **two-step methods** (PCA-UNet-Two, PCA-UNet-ATT-Two) perform similarly to **one-step methods** (PCA-UNet, PCA-UNet-ATT).
- For **MAE**, the **one-step method** (MAE-One) outperforms the **two-step method** (MAE-Two) across all four noise types.
- In the **UNet** family, **one-step methods** (UNet-One, UNet-One-256) perform better than the **two-step method** (UNet-Two) for *Real Image Noise*. However, all methods perform similarly for *Occlusion Noise* and *Thresholded Probability Noise*, while **two-step methods** excel in *Detection Image Noise*.

In terms of difficulty:

- *Real Image Noise* is the easiest to handle, followed by *Occlusion Noise*.
- *Thresholded Probability Noise* is more challenging, especially at higher noise levels.
- *Detection Image Noise* is the most difficult to manage in the wild environment.

Conclusion

The results reaffirm that modern, deep learning-based models, particularly **PCA-UNet**, which integrates PCA and U-Net architectures, are more capable of recovering accurate shapes from heavily corrupted inputs. This study not only provides a comprehensive comparison of current techniques but also offers insights for future improvements in robust shape denoising.

BIBLIOGRAPHY

Barbu, A. (2012). Hierarchical object parsing from structured noisy point clouds. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1649–1659.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522.

Borenstein, E., Sharon, E., and Ullman, S. (2004). Combining top-down and bottom-up segmentation. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 46–46. IEEE.

Bryner, D. and Srivastava, A. (2014). Bayesian active contours with affine-invariant, elastic shape prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 312–319.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017a). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017b). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

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

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338.

Flusser, J. and Suk, T. (1993). Pattern recognition by affine moment invariants. *Pattern recognition*, 26(1):167–174.

Grauman, K. and Darrell, T. (2005). The pyramid match kernel: Discriminative classification with sets of image features. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1458–1465. IEEE.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2021). Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Jalba, A. C., Wilkinson, M. H., and Roerdink, J. B. (2006). Shape representation and recognition through morphological curvature scale spaces. *IEEE Transactions on Image Processing*, 15(2):331–341.

Kendall, D. G. (1984). Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London mathematical society*, 16(2):81–121.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Kushner, H. and Yin, G. G. (2003). *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media.

Larrazabal, A. J., Martínez, C., Glocker, B., and Ferrante, E. (2020). Post-dae: anatomically plausible segmentation via post-processing with denoising autoencoders. *IEEE transactions on medical imaging*, 39(12):3813–3820.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE.

Long, C. and Barbu, A. (2022). A study of shape modeling against noise. In *ICIP*, pages 611–615. IEEE.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Pang, B., Han, T., Nijkamp, E., Zhu, S.-C., and Wu, Y. N. (2020). Learning latent space energy-based prior model. *arXiv preprint arXiv:2006.08205*.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174.

Pitas, I. and Venetsanopoulos, A. N. (1992). Morphological shape representation. *Pattern Recognition*, 25(6):555–565.

Raju, A., Miao, S., Jin, D., Lu, L., Huang, J., and Harrison, A. P. (2022). Deep implicit statistical shape models for 3d medical image delineation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2135–2143.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Rogers, M. and Graham, J. (2002). Robust active shape model search. In *European conference on computer vision*, pages 517–530. Springer.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455. PMLR.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science.

Sun, L., Wang, M., Zhu, S., and Barbu, A. (2024). A novel framework for online supervised learning with feature selection. *Journal of Nonparametric Statistics*, pages 1–27.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *NeurIPS*, 30.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology.

Yang, J., Liu, S., and Wang, X. (2021). Centered convolutional deep boltzmann machine for 2d shape modeling. *Personal and Ubiquitous Computing*, pages 1–11.

CHAPTER 8

BIOGRAPHICAL SKETCH

Cheng Long received his Bachelor of Science degree in Information and Computing Science from Sun Yat-Sen University, Guangzhou, China, in June 2010. He pursued his graduate studies at Florida State University, Tallahassee, USA, where he earned a Master of Science in Statistics in May 2020. Cheng is currently pursuing his Doctor of Philosophy in Statistics at Florida State University.

Throughout his academic journey, Cheng has engaged in a variety of professional roles and projects. His most recent experience includes an Applied Scientist Intern role at Microsoft, Redmond, USA, where he worked on the AI quality for Azure OpenAI Service: On-Your-Data, a Retrieval-Augmented-Generation (RAG) service, and contributed to the AI Evaluation Project, GPTAsserts. Prior to this, Cheng served as a Channel Manager and Risk Manager at the Bank of China, Guangzhou, China, where he developed systems to enhance operational efficiency and mitigate financial risks.