DEEP LEARNING FOR PRINTED IMAGE QUALITY

by

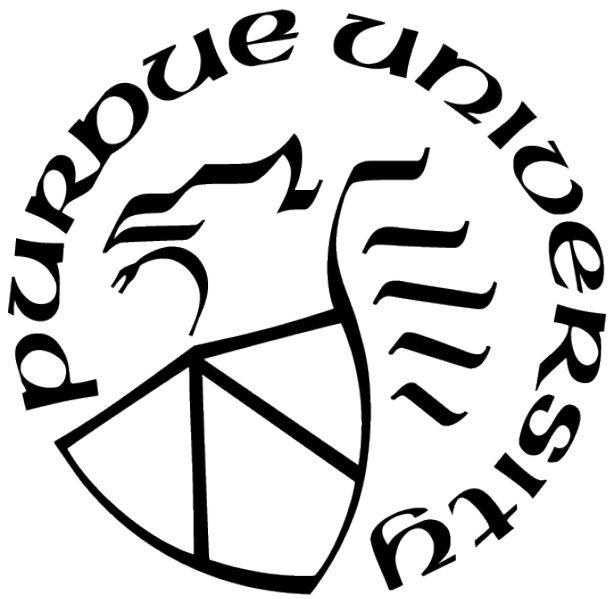
Jianhang Chen

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



School of Electrical and Computer Engineering

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Jan P. Allebach, Co-Chair

School of Electrical and Computer Engineering

Dr. Qian Lin, Co-Chair

School of Electrical and Computer Engineering

Dr. Fengqing Maggie Zhu

School of Electrical and Computer Engineering

Dr. Michael D. Zoltowski

School of Electrical and Computer Engineering

Approved by:

Dimitrios Peroulis

To my beloved parents.

ACKNOWLEDGMENTS

Research supported by HP Labs, Inc., Palo Alto, CA 94304.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DPI	Dot Per Inch
RGB	Red Green Blue
PQ	Print Quality
RANSAC	Random Sample Consensus
CNN	Convolutional Neural Networks
VGG	Visual Geometry Group
ResNet	Residual neural Network
GANs	Generative Adversarial Networks
MAP	Mean Average Precision
IoU	Intersection over Union
UNIT	UNsupervised Image-to-image Translation network
pix2pix	pixel-to-pixel network (Conditional Adversarial Networks)
VAEs	Variational Auto-Encoders
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
MSE	Mean Square Error
NCC	Normalized Cross Correlation
FIRE	Fundus Image Registration Dataset
VR	Virtual Reality
AR	Augmented Reality
MLE	Maximum Likelihood Estimator
EM	Expectation Maximization
VTN	Volume Tweening Network

ABSTRACT

This research focuses on developing algorithms to automatically classify, detect, simulate and improve the quality of defective printed images since the human visual system is unreliable. With the development of deep learning algorithms, state-of-the-art accuracy could be achieved for many computer vision tasks. This research applies the deep learning method to printed image quality assessment. Because most deep learning approaches require a large amount of data even after data augmentation, we propose to use Generative Adversarial Networks for simulation images generation. The simulated images with artifacts could be used for training classifier, detector and corrector networks for printed image quality. Another essential preprocessing step for printed image quality assessment is image registration, which can detect the defect and difference between two input images. This research proposes to use the deep learning framework for global image registration by parallel computation acceleration. For deformable local registration, we implement the U-Net VoxelMorph-based method for printed image registration. Then we further propose the recurrent network-based method, R-RegNet. The experimental results show that the proposed R-RegNet method outperforms the U-Net VoxelMorph-based method in all three datasets that we considered. Finally, we propose a photorealistic image dataset simulation method for training deep neural networks. A new dataset with simulated images, named Extra FAT, is introduced for object detection and 6D pose estimation.

1. INTRODUCTION

This research mainly focuses on developing the algorithms to classify, detect, simulate and improve the quality of printed images with different defects.

In Chapter 2, we introduce image quality assessment, including defect classification, grading and detection. The image quality assessment process is costly and time-consuming. The visual system of human beings is not always reliable in some cases, like the light/dark band optical illusion. There are some traditional defect classification and detection methods, but they are not accurate enough. With the development of deep learning algorithms, the state-of-art of accuracy could be achieved for many computer vision tasks. It is the first time to implement deep learning-based approaches in printed image defect classification and detection.

In Chapter 3, we introduce Generative Adversarial Networks (GANs) for printed image simulation. Most deep learning approaches require a large amount of data even after deploying traditional data augmentation algorithms. The Generative Adversarial Networks method is one of the generative models that could generate simulation images after learning from the samples. The simulated images could be used for training classifier, detector and corrector networks for printed image quality assessment.

In Chapter 4, we focus on the printed image registration problem. This research investigates global and local image registration. For global image registration, we propose to use the deep learning framework PyTorch for the intensity-based image registration algorithm. The SIMULATED dataset and HPLAB dataset with real printed pairs are collected for image registration. For local deformable image registration, we implement the U-Net VoxelMorph-based method for printed images and propose the recurrent network-based method, named R-RegNet. Experimental results prove the recurrent network-based method R-RegNet outperforms the U-Net VoxelMorph-based method in terms of mean square error or ground truth deformation error in all three test datasets that we considered.

In Chapter 5, a photorealistic image dataset simulation method is proposed for training deep neural networks. A new dataset with simulated images, named Extra FAT, for object detection and 6D pose estimation is introduced in this part.

2. DEEP LEARNING FOR PRINTED DEFECT CLASSIFICATION AND DETECTION

2.1 Introduction

Printed defects commonly appear on printed images, such as mottle, banding, drips, drop-density-differences, ghosting, folds, streaks, and smudges. Visual inspection is the primary approach to evaluating defect conditions. According to [1], there are two different methods for an operator to inspect the defects. First, an operator is fully occupied with one machine when the prints come off the press. Second, an operator could check a stack of prints instead. Thus, the operator does not need to wait; but the paper is wasted. However, human-based defect inspection is limited to qualitative evaluation, and is time-consuming. Since automated defect detection can address the limitations of human-based inspection, many researchers and companies have been attracted to developing a computer vision-based method for defect grading and detection [2]–[5].

2.1.1 Previous Work on PQ Defect Assessment

Early work on print quality (PQ) defect diagnosis reported a tool for computing the strengths of various PQ defects from scanned pages, based on procedures recommended by an ISO standard [6]. Tools to enable the customer to troubleshoot his or her print quality (PQ) issues by visual inspection were also developed. This work consisted of PQ troubleshooting pages [7], web-based troubleshooting tools [8], and tools for simulating the appearance of print quality defects on test pages [9]. For banding, in particular, tools were developed to model the defect and measure its strength via psychophysical experiments [10]–[13], as well as to estimate the period of periodic banding defects [14]–[16]. Some efforts focused on wavelets as a tool for PQ analysis [17]–[19]. Later efforts considered the visibility of PQ defects in the presence of customer content [1]–[3], [5], [20]–[25] and the identification of specific defects, such as mottle [26]–[31], macro-uniformity [32], fading [33], [34], ghosting [35], local nonuniformities [36]–[38], and streaks [39], [40]. Other efforts considered a more comprehensive set of PQ defects [41]–[44]. More recently, machine learning approaches (linear

regression and support vector regression) have been deployed to predict the visibility of PQ defects based on ground truth provided by human observers [32], [45], including the image quality ruler method [19], [46]–[48]. The most recent efforts have included the development of a comprehensive system for assessing a variety of PQ defects in customer content [49], including segmentation of the page into multiple regions of interest, according to the type of page content [50]. During the entire course of this time, several standards have been developed for assessing PQ [51]–[53].

In the ISO/IEC 24790 international standard [52], which is a revised version of ISO/IEC 13660 [53], a method for hardcopy image quality is introduced which uses a single high-pass filter that integrates many components in an area to a value. This method calculates the standard deviation of $2mm \times 2mm$ cells for an area larger than $20mm \times 20mm$. This method is helpful for qualitative comparison. In another paper [26], the tile method is modified and expanded to provide more discrete data, including different cell sizes, the average of density, standard deviation, the average of standard deviation and the standard deviation of standard deviation, which can be used for quantitative analysis for the mottle defect. Other traditional methods [26], [41] use the information of cluster, statistics, and wavelets for the mottle defect characterization problem. Traditional computer vision methods estimate the mottle defect by extracting visual features, such as frequency and standard deviation, from the image and characterize the defect level using a threshold based on experimental results. The main drawback of those approaches is that they need manually designed features for the printed defects.

2.1.2 Deep Convolutional Neural Networks

In recent years, the deep Convolutional Neural Network (CNN) has become one of the most efficient methods to solve many computer vision problems, such as segmentation [54], [55], detection [56], tracking [57], and pose estimation [58], [59]. With annotated data, the deep learning based-approach can treat the computer vision problem as a regression or classification task by extracting features in the CNN layers and joining them in fully connected layers. In contrast to the aforementioned traditional methods, the deep neural

network can be trained in an end-to-end manner based on the dataset without the necessity to manually design features and thresholds.

In most traditional methods, the mottle defect level depends on the local areas in an image with uniform color or grayscale. In the deep learning approach, the entire image can be used as the input to the network. The deep neural networks work on printed images with different and non-uniform contents. The traditional feature-based methods are limited to a specific defect. However, deep learning methods can easily be extended to different types of printed defects by training on different datasets.

This thesis proposes a deep learning-based method for printed mottle defect grading. We collected the training data scanned from printed pages and trained a ResNet-34 model to classify various images with different mottle defect levels.

2.2 Defect Image Classification

In this section, we introduce an algorithm to classify the defects of printed images. The network is trained for a classifier on two classes: Ghost VS Streaks. Ghost defect means the toner of an image from the printer is much lighter and less detailed than the digital image. Ghosts are often caused by a fault with the drum or fuser unit of the printer. The toner particles are not heated enough to produce a good image. According to [60], streak is a defect characterized by elongated blotches of ink in non-image areas, commonly caused by mechanical problems with the press, such as damaged rollers (in lithography), or by improper wiping of a gravure cylinder by the doctor blade (due to dried ink fragments attached to it, or other such problems).

In the dataset, there are 582 training images and 60 test images. For each image, the resolution is 7146×5146, as shown in Figure 2.1.

We use transfer learning [61] based on pre-trained ResNet34 and ResNext50 CNN [62] to solve the defect classification problem. Data augmentation methods including rotating, flipping, zooming are implemented for the original dataset, as shown in Figure 2.2. The Learning rates for different layers are $1e^{-4}$, $1e^{-3}$ and $1e^{-2}$ from lower level to higher level.

The accuracy for classification in ResNet34 is 60% and ResNext 50 is 65%. Figure 2.3 is the result of ResNext50 shown as a confusion matrix.

2.3 Defect Level Grading

Mottle is a printed defect caused by low-frequency random non-uniformity. The mottle defect differs from banding or any other periodic non-uniform defect that could appear at any gray level or color. Figure 2.4 shows an example where the mottle defect in certain areas severely degrades the printed image quality. From the perspective of human visual perception, the spatial frequency, size, contrast, sharpness, illumination, and viewing distance can affect the perception of the mottle defect. Figure 2.5 provides two visual examples of mottle defects, which are zoomed-in views from Figure 2.4.

In this work, we categorize the mottle defect level into four classes based on the defect degree:

1. Class A means the printed image is visually good. The uniform areas in the printed image look highly smooth and uniform.

2. Class B means the printed image is visually sufficient. There are some non-uniform blotches in the printed image.

3. Class C means a lack of printed image quality. There are some large non-uniform blotches in the image.

4. Class D means the printed image quality is inferior. There are huge non-uniform blotches.

2.3.1 Dataset

This section presents two new datasets used for mottle defect grading in printed pages. The first dataset is named the T dataset with 135 images with different levels of the mottle defect, as shown in Figure 2.6. In the T dataset, there are 22 images in class A, 63 images in class B, 45 images in class C, and 5 images in class D. All the images in the T dataset have

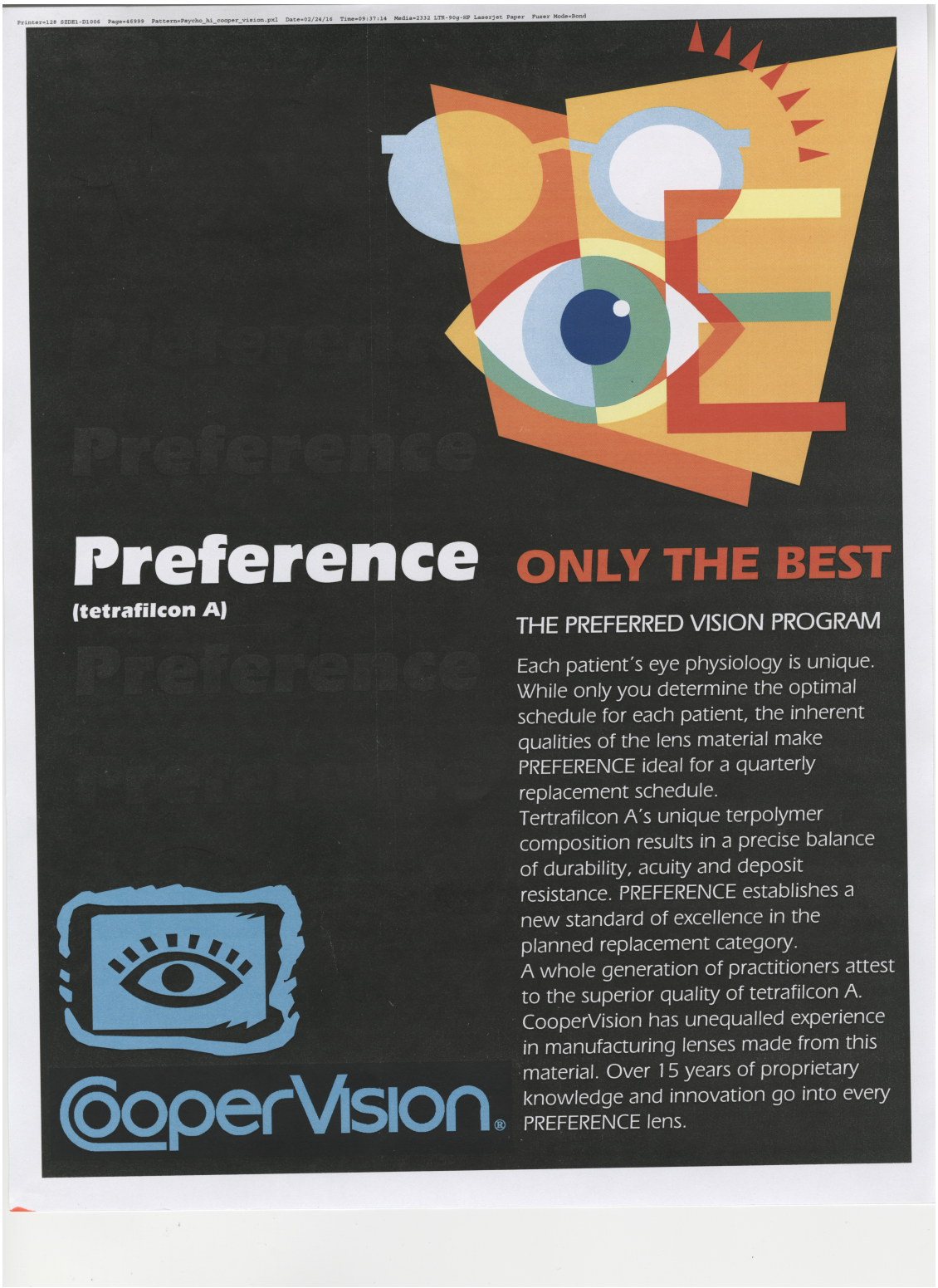


Figure 2.1. Example of a image in the dataset.

Figure 2.2. Example of data augmentation.

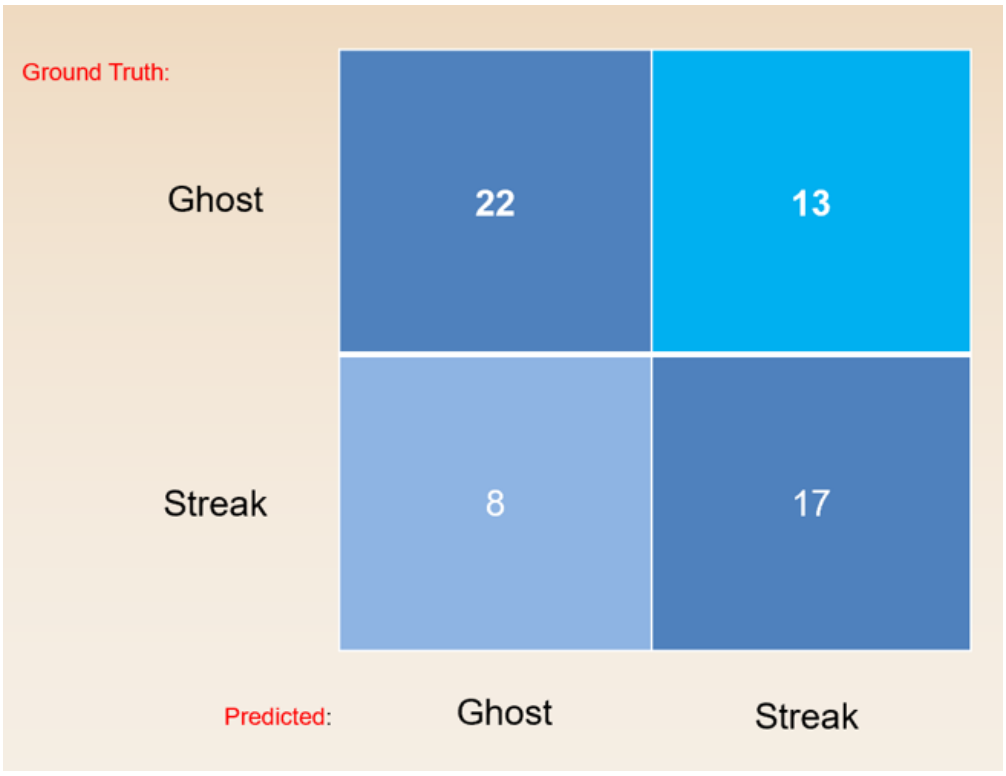


Figure 2.3. Confusion matrix of the ResNext training result.

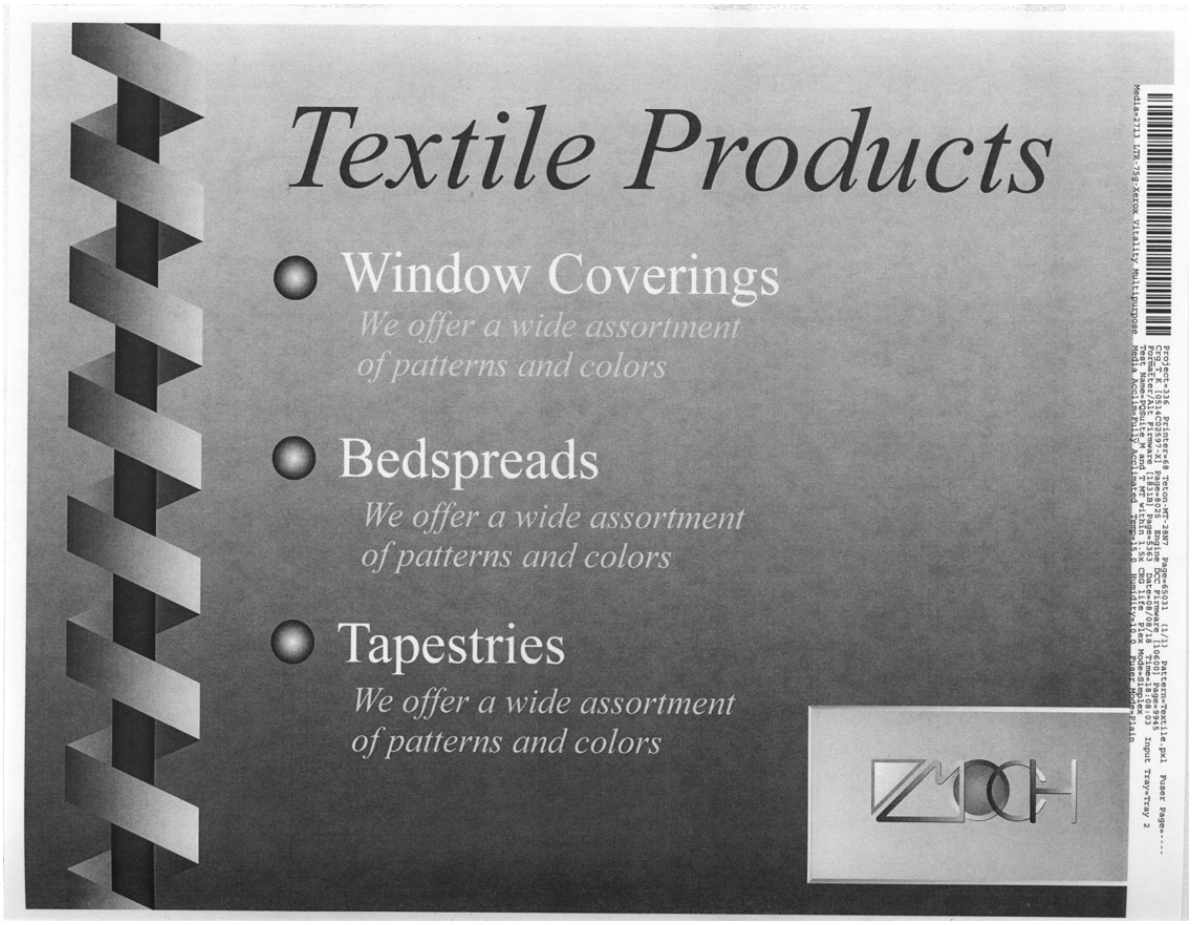


Figure 2.4. A printed image with mottle defect.

the same "textile" content. The second dataset is named the M dataset with 145 images as shown in Figure 2.7. In the M dataset, there are 87 images in class A, 42 images in class B, 15 images in class C, and 1 image in class D. The images in the M dataset have different contents. All images in the two datasets are grayscale and annotated by professional operators. The classes denote different quality levels ranging from the highest (A) to the lowest (D).

Data Augmentation and Unbalanced Datasets

The deep learning-based methods require a large amount of training data. To increase the dataset size in our work, we apply different data augmentation methods such as rotation, flip, zoom, and shift, as shown in Figure 2.8. Some commonly used filters, such as the Gabor

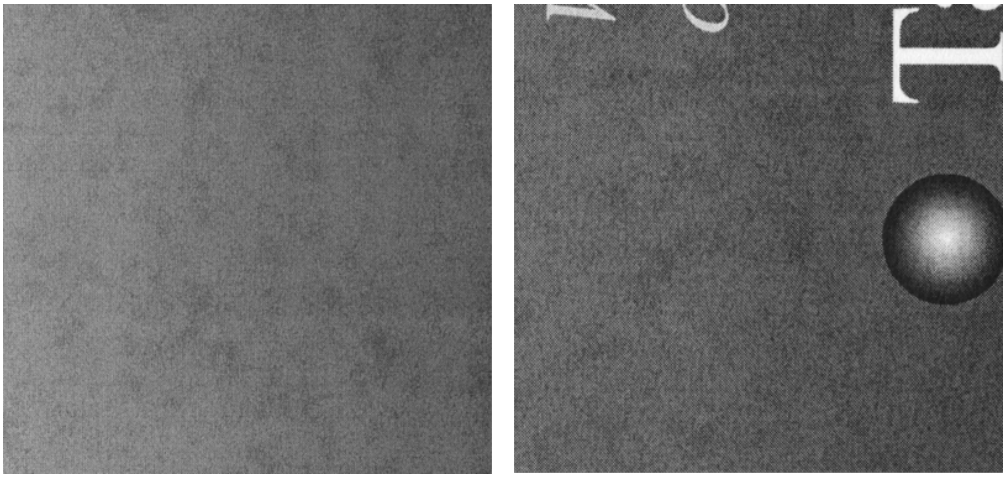


Figure 2.5. Zoomed in views of the mottle defect. The left image has lower density with uniform content. The right image has higher density with non-uniform content.

Figure 2.6. Examples from the T dataset which has 135 images with the same "textile" content.

filter and Gaussian filter that were used for augmentation of the ImageNet Dataset [63] are not used in this work because the filters might change the feature distribution of the mottle defects.

In both the T dataset and the M dataset, there are fewer samples in the D class, which corresponds to the poorest printed image quality. An unbalanced dataset like this may cause poor classification accuracy. Several methods have been developed to increase the number

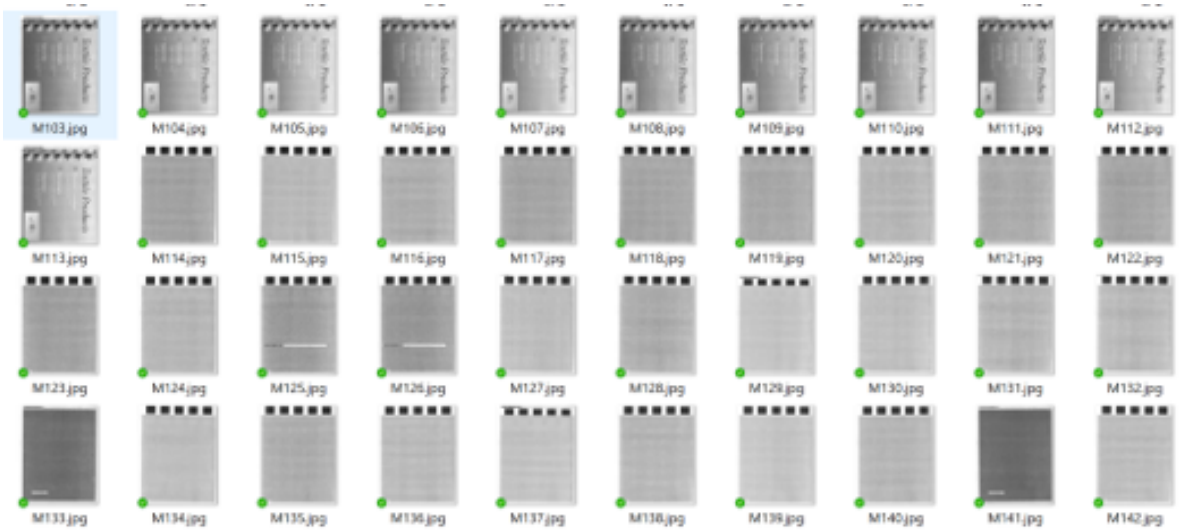


Figure 2.7. Examples from the M dataset which has 145 images with different contents.

Figure 2.8. Examples of data augmentation: a combination of augmentation methods such as rotation, flip, zoom, and shift are used in the above images.

of data for the classes with fewer samples. The first method uses a simulation method to generate synthetic data. It requires an accurate simulation model of the printing process and the mottle defect generation process, which are not available in our case. The second method randomly under-samples images from the original dataset. A higher sampling probability is assigned to the class with fewer samples to achieve balance. Although it is easy to implement, the under-sampling process eliminates some samples and loses essential information in the original dataset. The third method randomly over-samples images from the classes with insufficient samples. The maximum number of duplicates is limited to avoid overfitting in the training dataset.

In this work, the images in classes with insufficient data are over-sampled; and the maximum number of duplicates is limited to 15. Then, the data augmentation is implemented for the training dataset. In the final dataset, the T dataset has 192 images with 44 images in class A, 63 images in class B, 45 images in class C, and 40 images in class D. The augmented T dataset is divided into a training set with 154 images and a validation set with 38 images. The T dataset and the M dataset are merged to generate the combined dataset. After augmentation, the combined dataset has 410 images with different contents. There are 109 images in class A, 105 in class B, 106 in class C, and 90 in class D. The augmented combined dataset has 328 images in the training set and 82 images in the validation set. The resolution of all images is 5100×6600 (600 DPI).

2.3.2 Network and Training Process

The ResNet-34 [62] network is used as a feature extraction backbone. The ResNet-34 structure has several residual blocks, as shown in Figure 2.9. The feature extraction network is followed by fully connected layers.

The regression network and classification network have similar network structures. The main difference is the last fully connected layer and output of the network. In the regression network, the last layer has a single output, as shown in Figure 2.10. The four classes $[A, B, C, D]$ are mapped to scores $[0, 1, 2, 3]$. In the classification network, the last fully connected layer has four outputs, as shown in Figure 2.11.

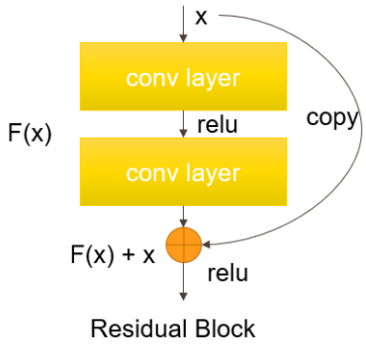


Figure 2.9. Structure of the residual block.

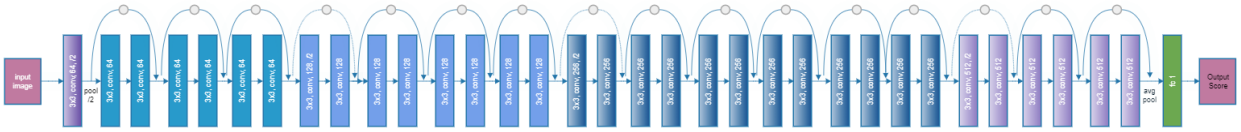


Figure 2.10. Network structure for the mottle defect regression task.

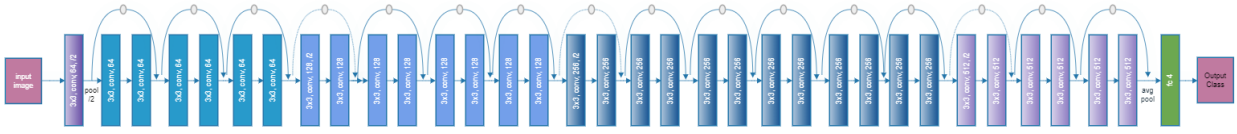


Figure 2.11. Network structure for the mottle defect classification task.

The final network is trained by transfer learning using ResNet-34 pre-trained on ImageNet [63] connected with fully connected layers. In the first step, the ResNet-34 convolution layers are frozen. Second, the fully connected layers are trained with a specified learning rate. Third, the convolution layers are unfrozen and the whole network is retrained with a lower learning rate. In the training process, the loss function is MSE Loss adding L2 parameter regularization loss for the regression task. The loss function for the classification task is the cross-entropy loss adding L2 parameter regularization loss. In both regression and classification tasks, we use dropout [64] in the training stage. Dropout is a technique that randomly removes some nodes in neural networks with a probability p during training. By doing this, the network is more robust to the change of weighting for some specific nodes and can avoid overfitting. The probability p is also named dropout rate. In the validation/testing stage, the dropout and L2 parameter regularization loss is turned off.

To find the best learning rate for training, we first train the stochastic gradient descent [65] with a lower learning rate in each epoch. Then the learning rate is multiplied by a factor in each mini-batch until a higher learning rate is reached. We record the loss in each iteration for different learning rates and choose the learning rate with a relatively lower loss. The triangular learning rate policy [66] is used in the training stage. The learning rate value changes between the minimal and maximal learning rates. The increase of the learning rate will force the network model to explore a new parameter space when the loss function decreases slowly or stops decreasing.

2.3.3 Experimental Results

In this work, the deep CNN is trained on two different datasets, including the augmented T dataset and the augmented combined dataset.

The first experiment is trained on the augmented T dataset. In the first experiments, we train the dataset using cropped input images and combine the results of the small patches to generate the final prediction. We treat the defect grading problem as a regression or a classification problem to explore the effect on the prediction accuracy.

The second experiment is performed on the augmented T dataset. We train the network using the 154 images (without cropping) in the training set and validate using 38 images in the validation set. We also test both regression and classification methods in the second experiment.

The second experiment is performed on the augmented combined dataset. In the third experiment, we train both the regression and classification networks using the 328 images (without cropping) in the training set and validate using 82 images in the validation set. The combined dataset results in experiment three further prove that the network can be generalized to mottle defect grading with different contents.

Prediction using Cropped Patches

In the first experiment, the original input image in the T dataset is cropped to 600×600 resolution patches without overlap. The whole T dataset is cropped into 11880 patches in four classes: $[A : 22 \times 88, B : 63 \times 88, C : 45 \times 88, D : 5 \times 88]$. The training set has 9504 images and the validation set has 2376 images, respectively. The error rate on the training set is 38.55%. The confusion matrix is shown in Figure 2.12.

In the prediction stage, the original image is cropped to 88 patches as shown in Figure 2.13. The 88 patches are fed to the convolutional neural network to output a 88 dimension score vector. The final prediction is generated by averaging or majority voting over the score vector. On the validation set, the error rate is 33.33% by the average score method and 37.04% by the majority voting method. The result shows that the accuracy of using local cropped patches is not satisfactory.

Mottle Grading Regression and Classification on T dataset

The second experiment is performed on the augmented T dataset. In the regression task, the input is resized to 1000×1000 resolution and the output is a single score mapped to four classes [A, B, C, D]. We first use the learning rate finding as shown in Figure 2.14, which suggests a learning rate in the range of 1e-2 to 2e-1. The learning rate finding is to use a small percent of data for training and get minimum loss with different learning rates.

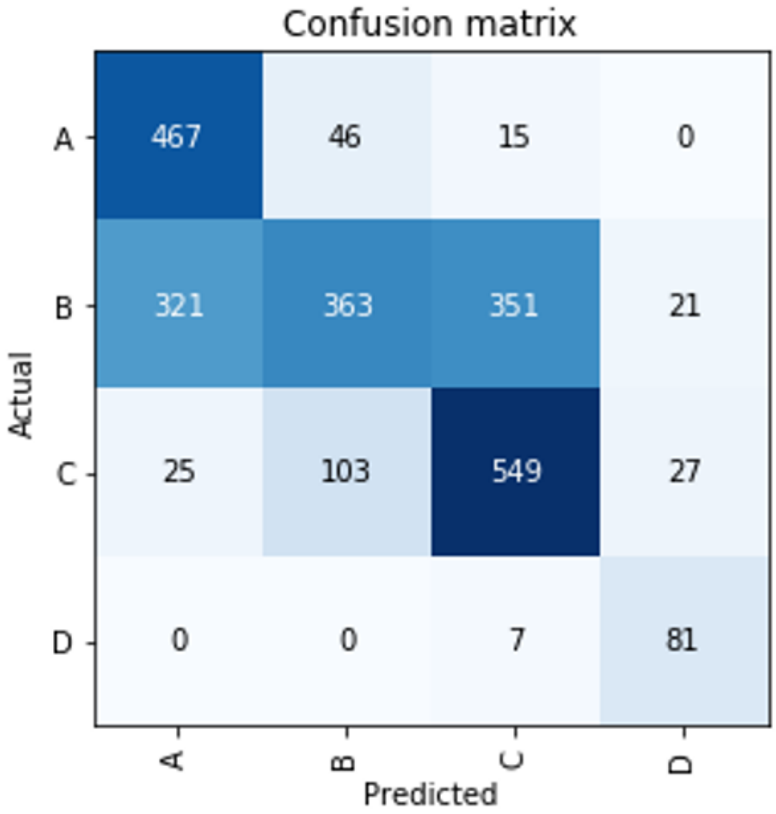


Figure 2.12. Confusion matrix for the prediction using cropped patches. The x-axis is the prediction of the networks and the y-axis is the ground truth of the data.

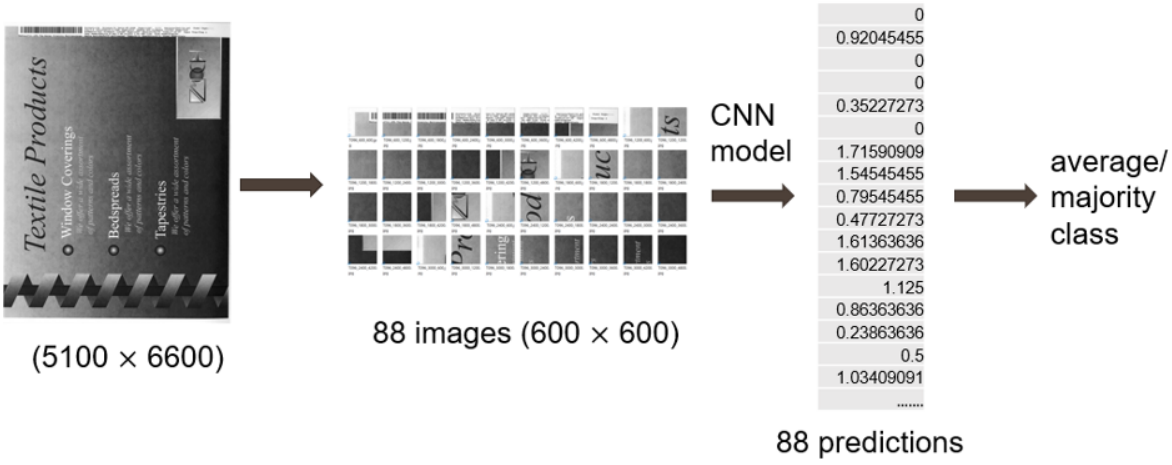


Figure 2.13. Defect prediction using cropped patches.

We then test different learning rates in the range of 1e-2 to 2e-1 in the whole dataset. The final best learning rate is 2e-2. Then we freeze the convolutional layers, unfreeze the fully connected layers, and train for 40 epochs. Finally, we unfreeze all layers and train with a lower learning rate in the range [3e-5, 2e-2] for 5 epochs. The batch size is 16 in the training process. Figure 2.15 shows the loss in the training and validation sets. The best root mean squared error (RMSE) for regression is 0.45, as shown in Figure 2.16. Here the RMSE is calculated as $RMSE = \sqrt{\frac{1}{n}\Sigma_{\mathrm{i}=1}^{n}\left(y_{\mathrm{i}} - \tilde{y}_{\mathrm{i}}\right)^2}$, where $\tilde{y}_{\mathrm{i}}$ is the predicted value from the regression model, y_{i} is the ground truth and n is the total number of tests. The final error rate is calculated as $\frac{\mathrm{e}}{n}$, where e is the number of false predictions. We round the predicted value $\tilde{y}_{\mathrm{i}}$ to $\bar{y}_{\mathrm{i}}$. When $\bar{y}_{\mathrm{i}} \neq y_{\mathrm{i}}$, it is a false prediction. The final error rate is 21.05% in the first experiment.

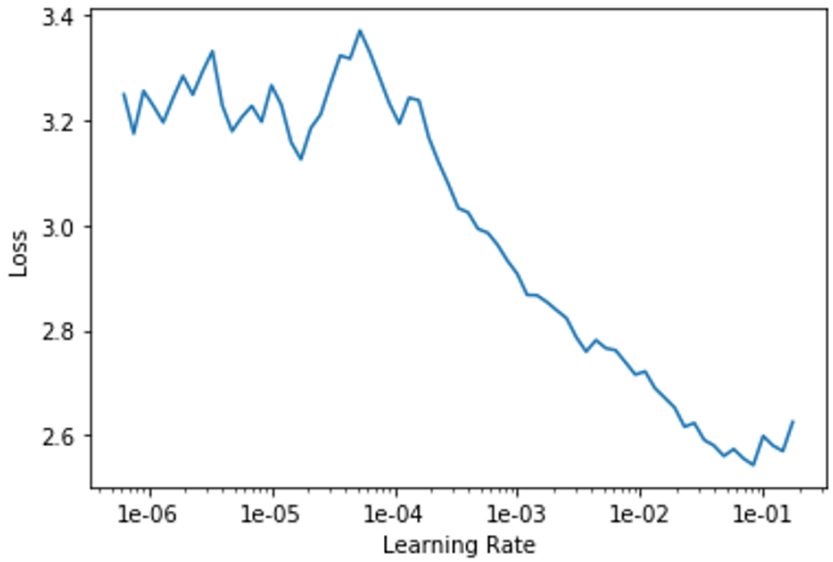


Figure 2.14. Finding the learning rate for the regression method on the T Dataset.

The last fully connected layer has four outputs indicating four classes in the classification task. The loss for the classification task is shown in Figure 2.17. The confusion matrix is shown in Figure 2.18. The error rate is 13.16%, which is better than the regression task in the T dataset with the "textile" content images.

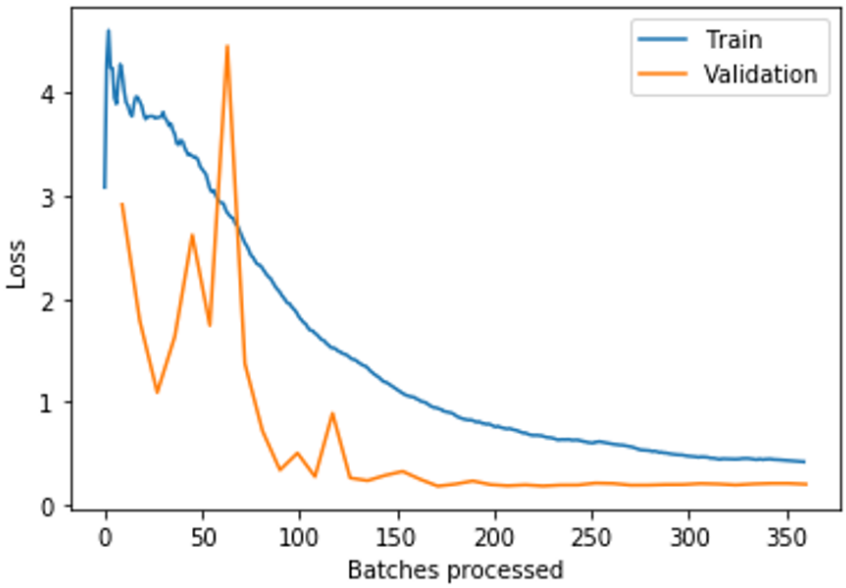


Figure 2.15. Loss for regression on the T training and validation sets. The dropout rate in the training stage is 50%. The loss in the training stage is calculated by adding MSE loss and the regularization loss of the weights. The loss in the validation stage is MSE loss only.

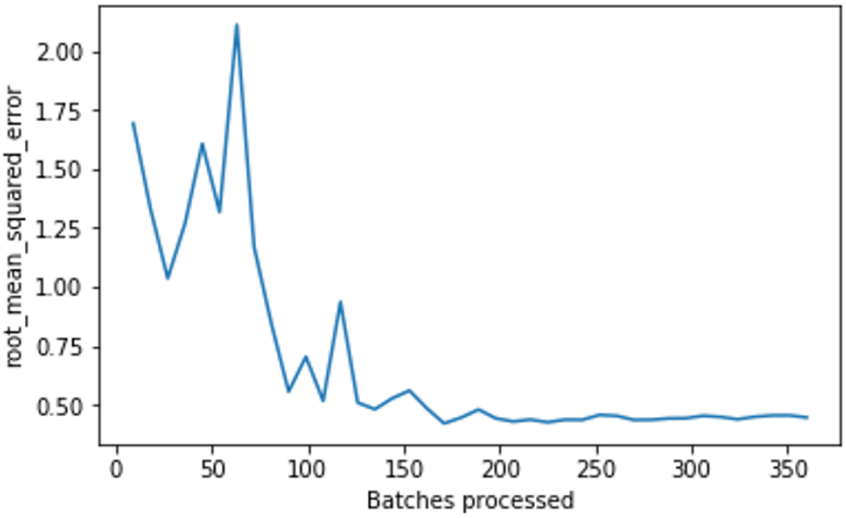


Figure 2.16. Root mean squared error for regression on the T dataset.

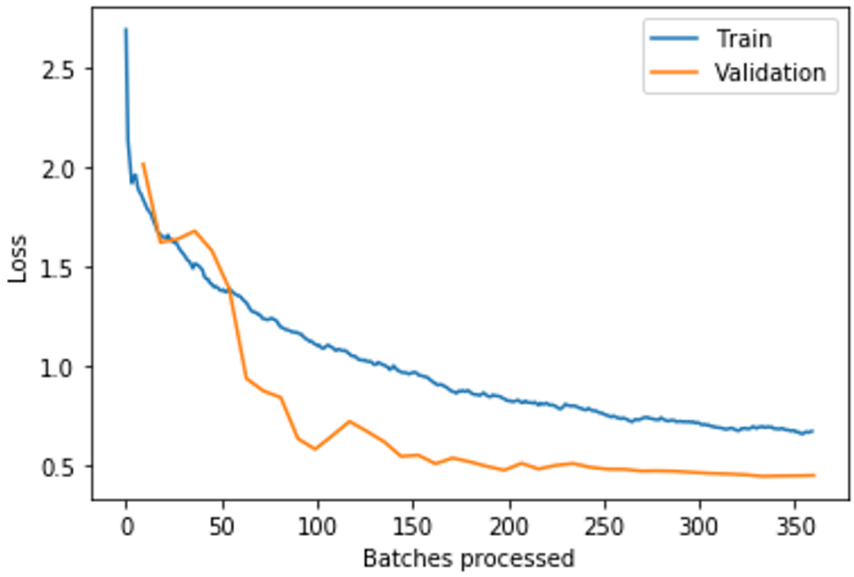


Figure 2.17. Loss for classification in the T training and validation sets. The dropout rate in the training stage is 50%. The loss in the training stage is calculated by adding cross-entropy loss and the dropout regularization loss of the weights. The loss in the validation stage is cross-entropy loss only.

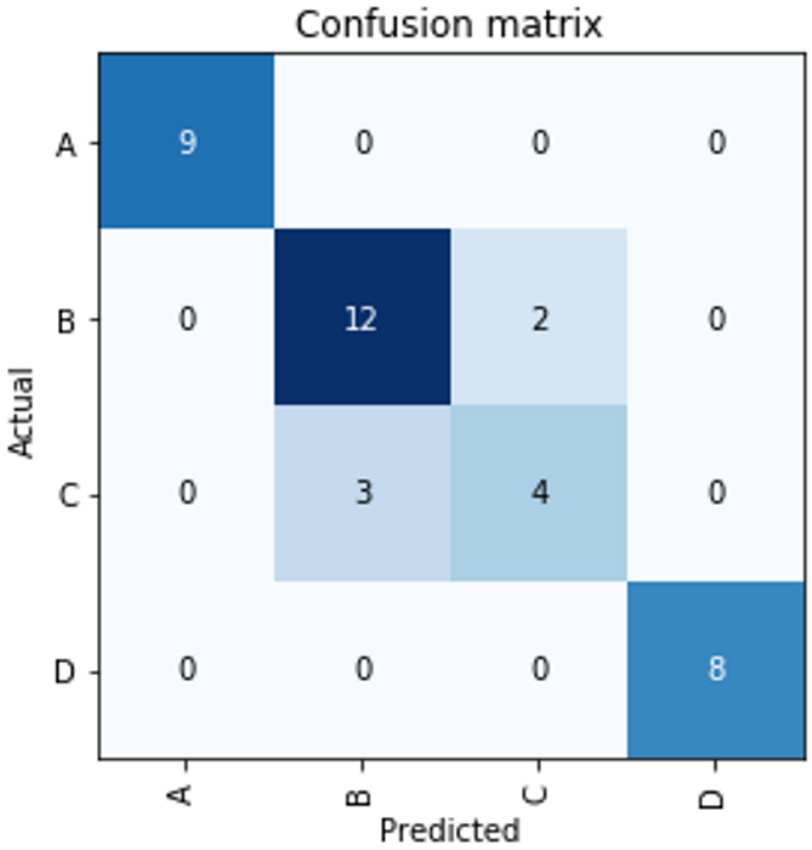


Figure 2.18. Confusion matrix for classification on the T dataset. The x-axis is the prediction of the network, and the y-axis is the ground truth of the data.

Mottle Grading Regression and Classification on the Combined Dataset

The second and third experiments are performed on the combined dataset. Mottle grading regression and classification on the combined dataset have a setting similar to that of the tasks on the T dataset. The combined dataset is more challenging since it has a large variety of image contents. In the regression task, Figure 2.19 shows the loss in the training and validation sets. The best root mean squared error for regression is 0.49, as shown in Figure 2.20. The final error rate is 21.95%. In the classification task, the loss for classification is shown in Figure 2.21. The confusion matrix is shown in Figure 2.22. The best error rate is 20.73% which is similar to the regression task.

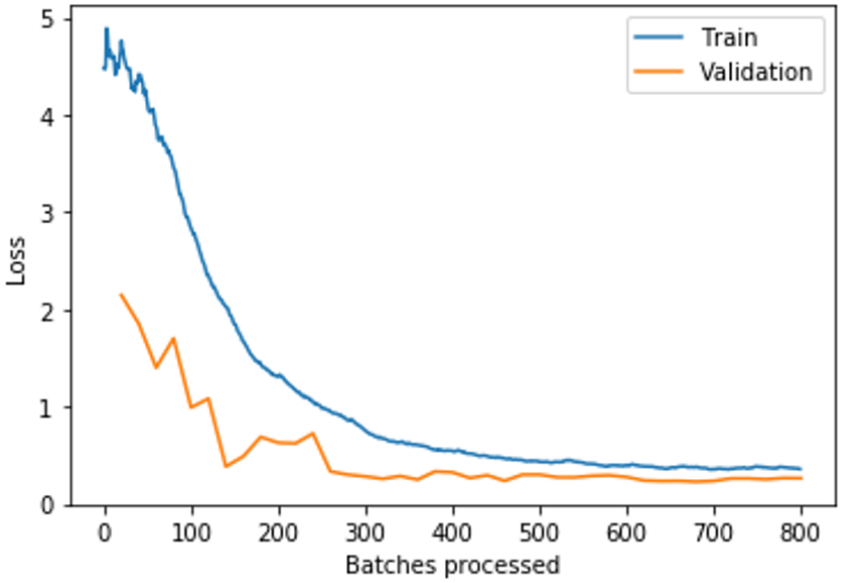


Figure 2.19. Loss for regression in the combined training and validation sets. The dropout rate in the training stage is 50%. The loss in the training stage is calculated by adding MSE loss and the regularization loss of the weights. The loss in the validation stage is MSE loss only.

2.4 Defect Detection for Printed Image

In real applications for printed images with defects, industries are more interested in detecting the location of the defective part of printed images. For the defects detection task,

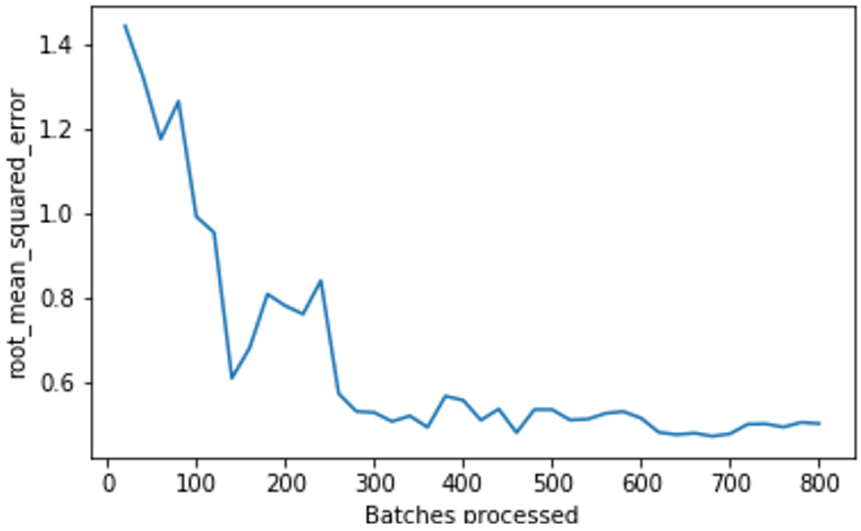


Figure 2.20. Root mean squared error for regression on the combined dataset.

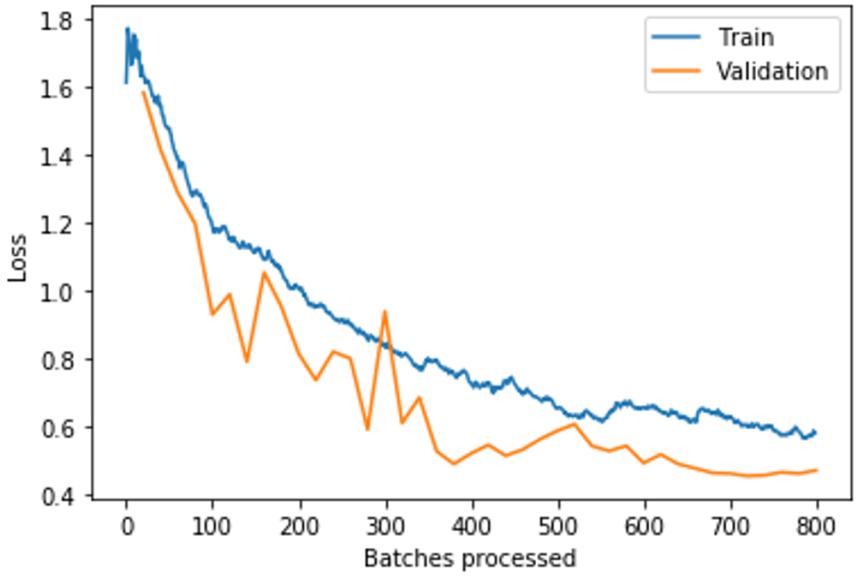


Figure 2.21. Loss for classification in the combined training and validation sets. The dropout rate in the training stage is 50%. The loss in the training stage is calculated by adding cross-entropy loss and the regularization loss of the weights. The loss in the validation stage is cross-entropy loss only.

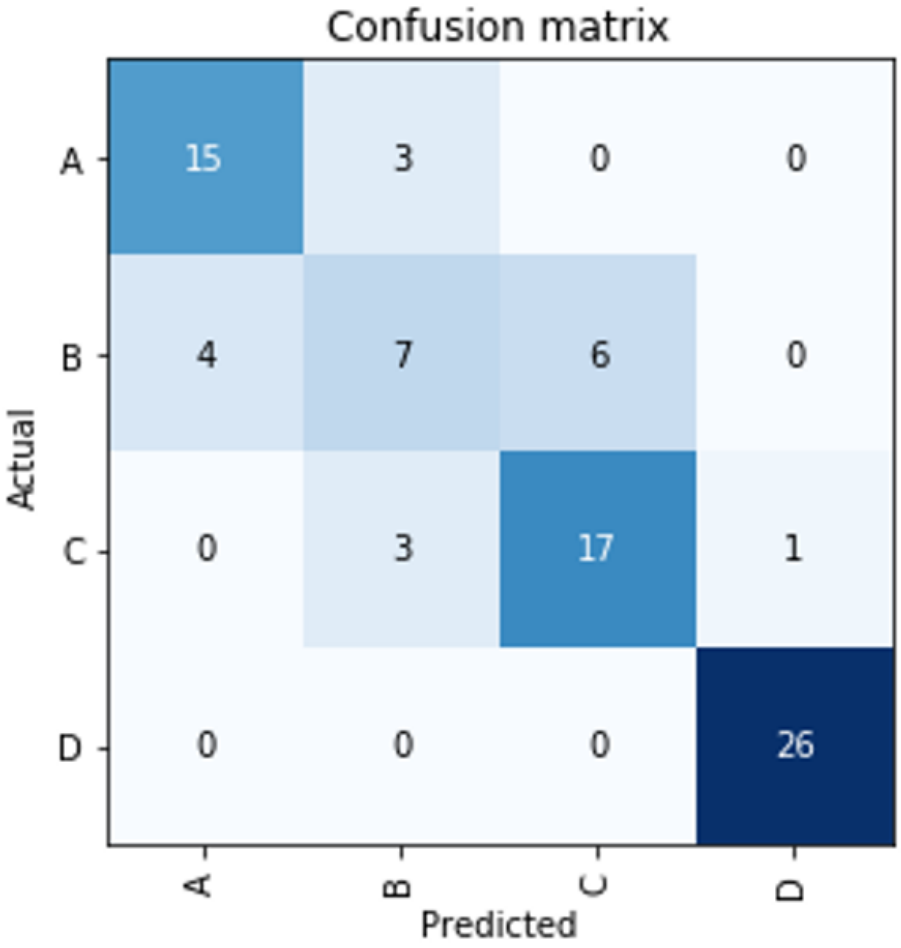


Figure 2.22. Confusion matrix for classification on combined dataset. The x-axis is the prediction of the network, and the y-axis is the ground truth of the data.

preliminary data could consist of one digital and one printed image in a pair or the printed image only. The printed images may have defects such as banding, bright streak, dark streak, wrinkle, etc. The defect object detection algorithm should be able to classify the category of the defect and provide a bounding box or map of the defective area. Although both digital and printed images could be given as the input of the algorithm, ideally, the goal for the algorithm is to detect defects with only input from scanned printed images. There are few works of literature about printed defect detection methods based on deep learning. However, defect detection algorithms based on CNN are used in other industries like civil engineering. For example, Naive Bayes data fusion and CNN [67] are used in crack detection and Faster R-CNN is used in damage detection [68], [69].

In the experiment, we mainly focus on the streak detection problem. The training dataset consists of 1100 images with streak defects in different locations. The test dataset has 100 images. Each image has bounding boxes to indicate the region of streaks. We implement a defect detection algorithm using Faster R-CNN [56] with a ResNet50 feature extraction network. As shown in Figure 2.23, the input image is transferred to features by the feature extraction network. Then the region proposal network proposes multiple regions of interest (ROIs). After the ROIs pooling stage, the feature of each ROI is fed into the classifier, which generates the final bounding box position. The Faster R-CNN network can be trained on Caffe and Caffe2 frameworks on 1 to 4 Titan Xp GPUs. The batch size per image is 256. For the input image, the original resolution is 7146×5146. As shown in Table 2.1, the images are rescaled to various resolution including 500×500 and 1000×1000. The best Mean Average Precision (MAP) is 0.23. In this experiment, the MAP is calculated for images with Intersection over Union (IoU) $>= 0.50$. The calculation of MAP is explained in the COCO dataset paper[70] in detail.

Figure 2.24 is an example of the result of streak detection on the test dataset. The green bounding boxes indicate the predicted region of the streak from our detection algorithm. We also test the trained defect detection model on a different dataset which has the images printed from total different models of printers. As shown in Figure 2.25, the yellow-colored streak is much wider compared to the original dataset, but the network could still detect the streak. The result shows that the streak detection algorithm works on a different dataset.

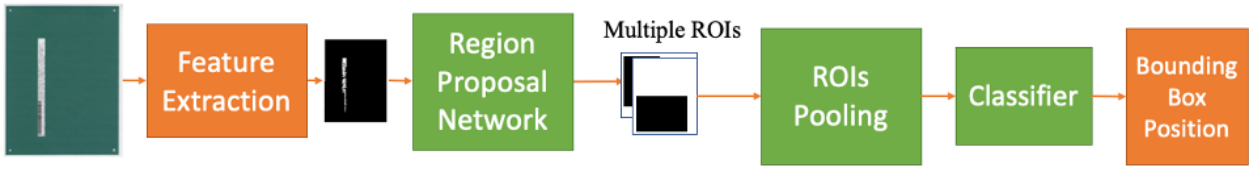


Figure 2.23. Structure of the Faster R-CNN network.

Table 2.1. Specifications of Faster R-CNN defect detection

GPU	Resolution	Rescale	MAP	Iteration Time	Inference Time
1	2624×1946	500	0.167	0.188-0.231s	0.057-0.100s
1	2624×1946	1000	0.23	0.225-0.340s	0.090-0.205s
1	7146×5146	1000	0.23	1.083-1.181s	0.132-0.230s
4	2624×1946	500	0.167	0.181-0.188s	0.047-0.054s
4	2624×1946	1000	0.23	0.229-0.236s	0.091-0.098s
4	7146×5146	1000	0.23	1.017-1.024s	0.098-0.105s

2.5 Conclusion

In this part, we implement CNN for solving the defect classification problem, printed mottle defect grading and Faster R-CNN for the defect detection problem. The main contribution of this work is to propose a deep learning-based method instead of traditional feature-based methods for printed image quality assessment. For printed mottle defect grading, we propose a new deep learning-based method. Unlike traditional methods such as feature extraction using ΔE variation, our method utilizes a CNN for the first time to automatically extract the feature by stochastic gradient descent. Transfer learning and data augmentation methods are used to train a robust mottle defect grader. The proposed deep learning mottle characterization method can be used in mottle grading not only for the test image with the same uniform content as seen in the training set but for printed images with different contents. The mottle grading method achieves a 13.16% error rate in the T dataset with the same content and a 20.73% error rate in the combined dataset with different contents. The proposed method can also be generalized to other printed defects, such as streaks

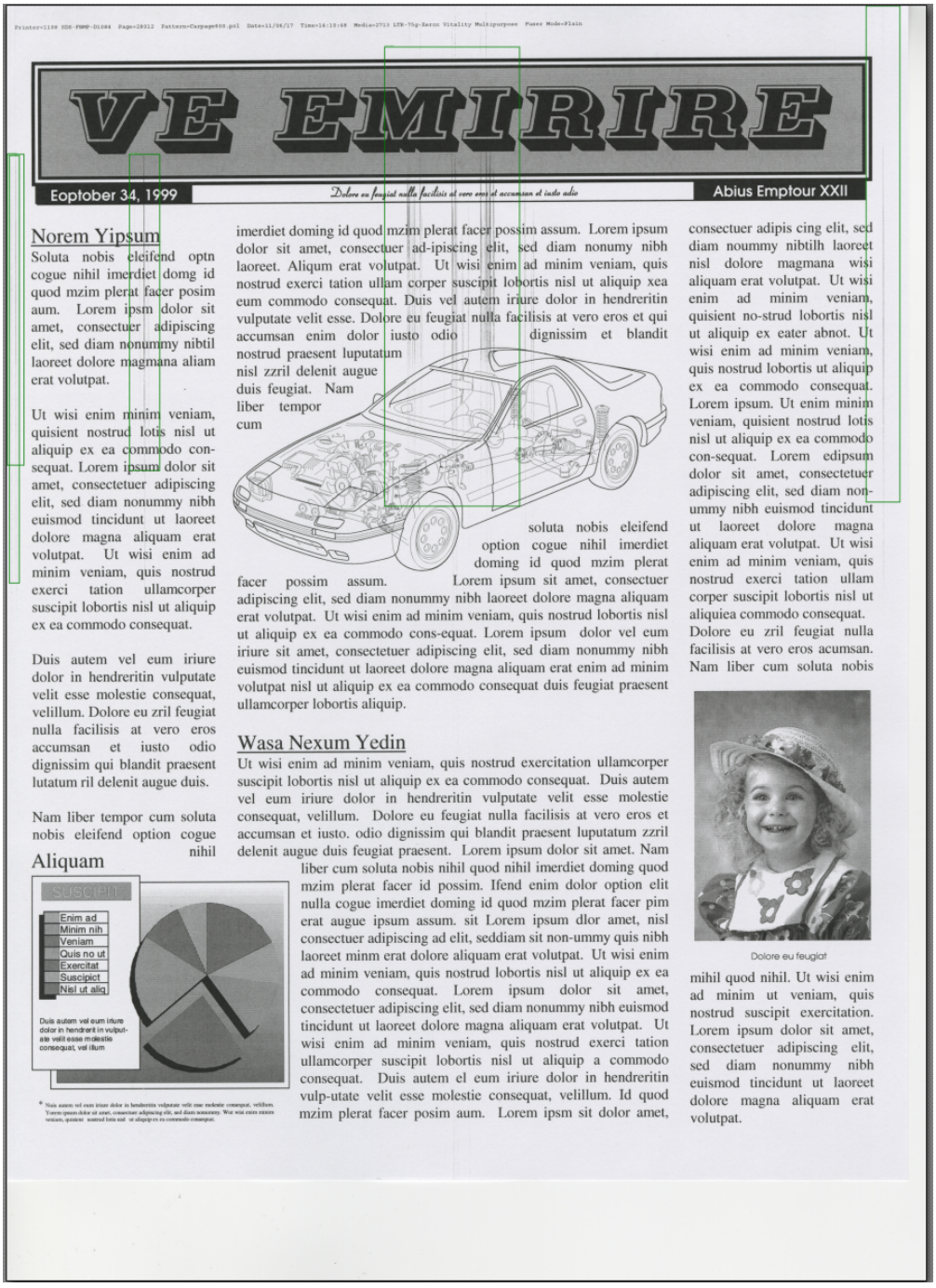


Figure 2.24. Example of streak detection.

Figure 2.25. Example of streak detection in another dataset.

given an annotated streak dataset. The results prove the feasibility of deep learning methods for defects from different domains.

3. GENERATIVE ADVERSARIAL NETWORKS FOR PRINTED IMAGE SIMULATION

3.1 Introduction

3.1.1 Dataset and Data Augmentation

In recent years, CNN has become one of the most efficient methods to solve computer vision problems such as segmentation, detection, tracking and classification. The previous chapter proposes deep learning methods for printed image defect classification and detection. However, deep learning methods based on CNN require a large amount of image data scanned from the printed images for printed image quality assessment. The accuracy and success of the deep neural network have been highly attributed to big datasets such as the ImageNet Dataset [63] and the COCO Dataset [70]. The quality, quantity and diversity of a dataset primarily affect the performance and generalization of the trained network.

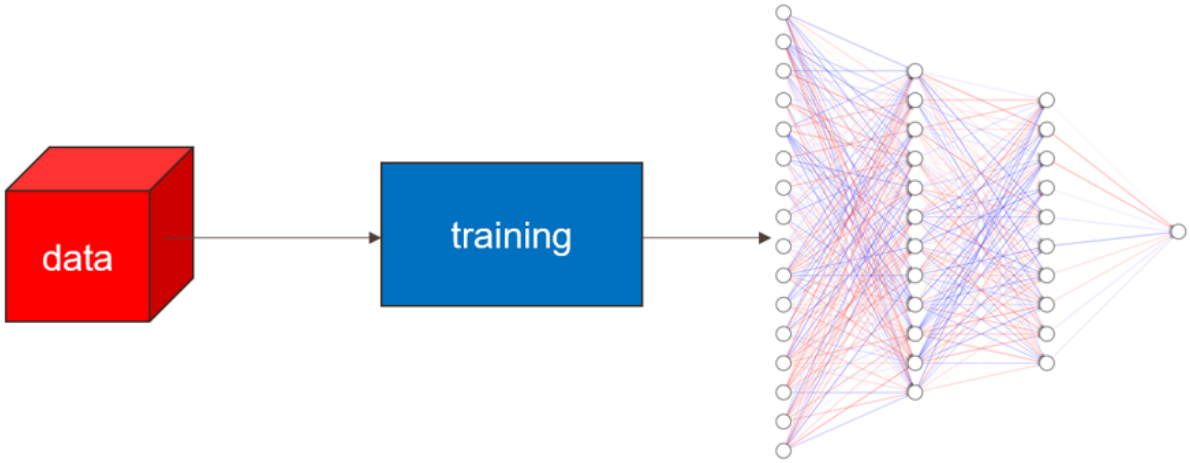


Figure 3.1. Training a deep neural network requires plenty of data.

Data augmentation is growing increasingly popular in machine learning and deep learning communities. It refers to techniques for generating samples by transforming training data, with the target of improving the accuracy and robustness of the networks without actually collecting data [71]. There are some common strategies such as flipping, rotation, cropping,

filtering and zooming for network training. Figure 3.2 shows examples of traditional data augmentation.

However, the images can only be slightly changed by those basic data augmentation methods, as shown in Figure 3.3. The quantity of data is still limited to a number proportional to the original dataset. In our case of the printed image dataset, we still have to print and scan images in order to collect data. Those printed images are hard to generate, as shown in Figure 3.4. Commonly, the labor required for image printing and scanning is intensive to generate the dataset. It is of value if we could find the essence of printed images and reproduce them directly from digital images using a model without the printing and scanning process.

3.1.2 Generative Model and Generative Adversarial Network

To simulate sample images from digital images requires an image generation model. One of the classical approaches is the Maximum Likelihood Estimator (MLE) [72] which chooses a pre-defined model and estimates its parameters through maximizing the likelihood of training data, as shown in Equations (3.1) to (3.4).

$$\theta^* = \arg\max_{\theta} \prod_{\mathrm{i}=1}^{n} p_{\mathrm{model}}(x^{(\mathrm{i})}; \theta) \tag{3.1}$$

$$= \arg\max_{\theta} \log \prod_{\mathrm{i}=1}^{n} p_{\mathrm{model}}(x^{(\mathrm{i})}; \theta) \tag{3.2}$$

$$= \arg\max_{\theta} \prod_{\mathrm{i}=1}^{n} \log p_{\mathrm{model}}(x^{(\mathrm{i})}; \theta) \tag{3.3}$$

$$\theta^* = \arg\max_{\theta} \mathbf{E}_{x \sim p_{data}} [\log p_{\mathrm{model}}(x^{(\mathrm{i})}; \theta)] \tag{3.4}$$

where θ is the parameters of the pre-defined model, and $x^{(\mathrm{i})}$ is a sample (i) from the dataset. By estimation algorithms such as the Expectation Maximization (EM) algorithm [73], maximum likelihood estimators could be calculated recursively from incomplete data.

(a) Original image.

(b) Image after vertical flipping.

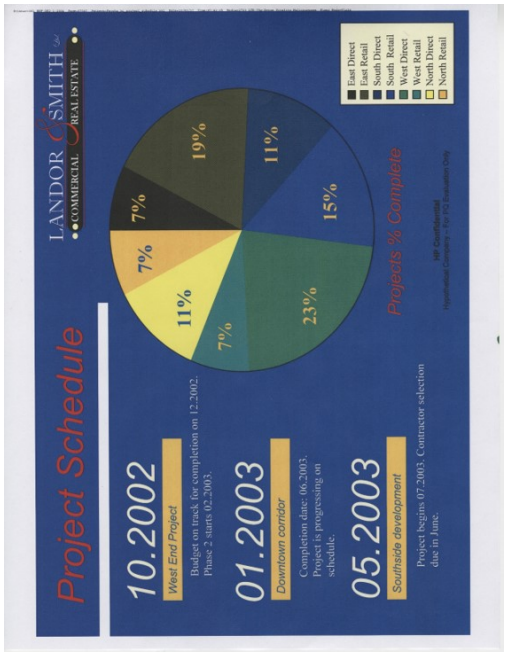
(c) Image after horizontal flipping.

(d) Image after rotation.

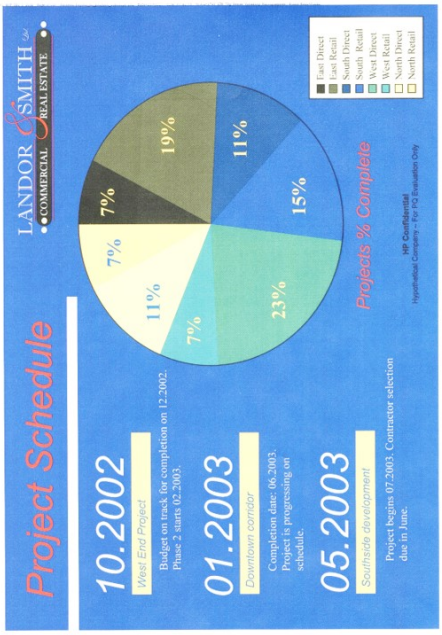
(e) Image after sharpening filter.

(f) Image after zooming.

Figure 3.2. Sample images of traditional data augmentation.

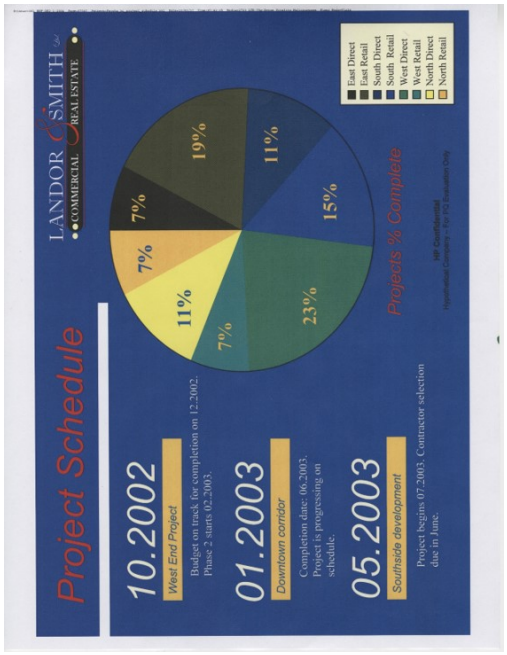


(a) Original image.

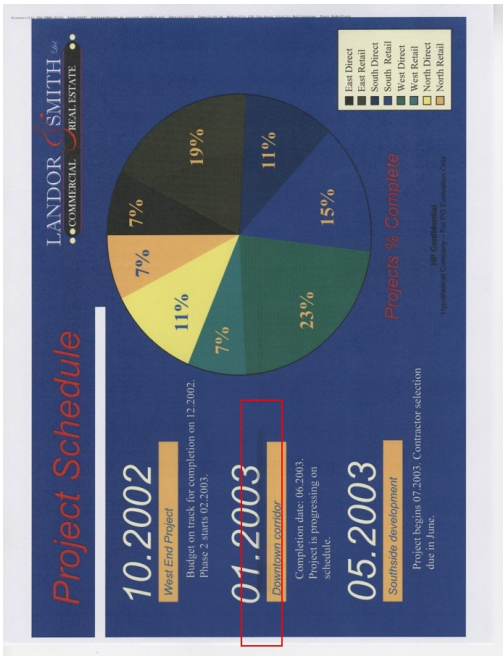


(b) Image changed in brightness and contrast.

Figure 3.3. Some training images can be easily generated.



(a) Original image.



(b) Printed image with streak.

Figure 3.4. Some training images are hard to generate.

However, explicit models like MLE require a well-defined likelihood function such as a truncated Poisson distribution or Gaussian distribution. It is hard to find unknown data-generating distribution and perform density estimation. There are other explicit models used in sample generation such as Variational Auto-Encoders (VAEs) [74], PixelRNN [75] and PixelCNN [76]. For example, VAEs use the posterior $p_\theta(z|x)$ to estimate the distribution between an image distribution $p(x)$ and a latent space z, where θ is the parameter of the distribution. However, simulated images from VAEs tend to be blurry because the approximation paradigm, which is often an unimodal Gaussian distribution, is oversimplified for the complex distribution of real images. The results of explicit models have lower quality compared to the state-of-the-art Generative Adversarial Networks (GANs) [77] which are implicit models.

GANs generate new samples for the model by using a loss function which calls another discriminator model. There is no need to find an explicit density function and maximize its likelihood or lower bound. Instead, GANs use the Nash Equilibrium concept from game theory. In GANs, there are two players: generator (player G) and discriminator (player D). Instead of using maximum likelihood estimator or optimizing a fixed loss function, GANs take advantage of the discriminator network to learn a dynamic loss function for the generator network. Thus, more realistic images could be generated from GANs through its adversarial process. On the other hand, the dynamic process of GANs is often harder to train with two sub-networks and sometimes suffers mode collapse.

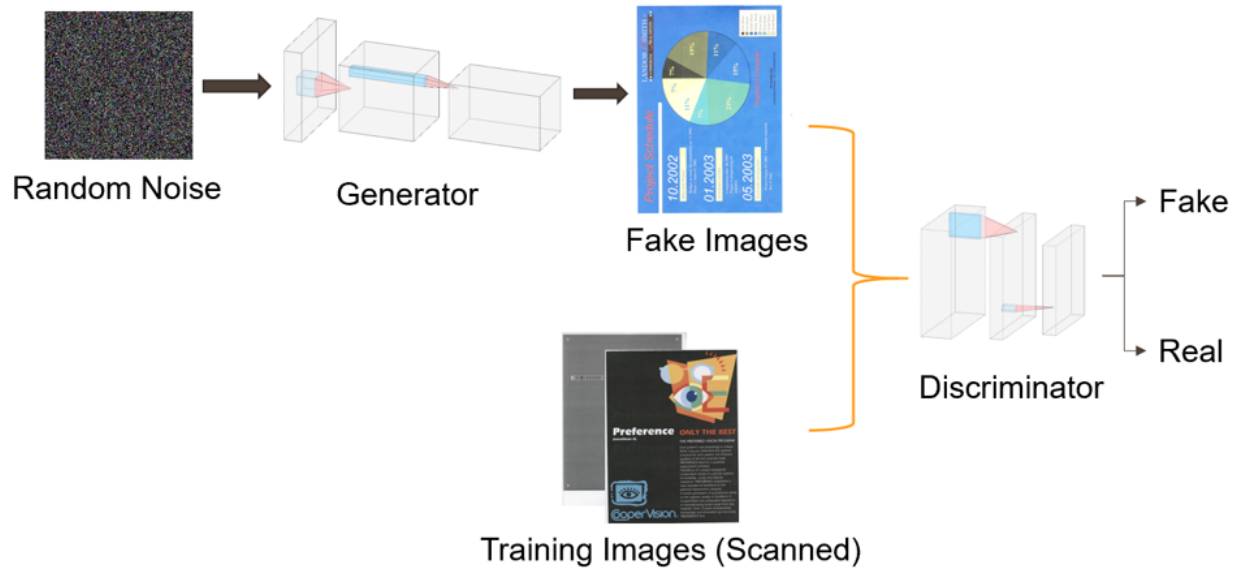


Figure 3.5. Generative adversarial networks via Nash Equilibrium.

As shown in Figure 3.5, random noise is generated as the input to the generator. The generator is trying to produce fake data as a counterfeiter. Respectively, the discriminator is trying to detect fake data produced by the generator. The discriminator is a classification network to distinguish between real and fake data. The competition between the generator and discriminator drives both networks to improve their accuracy. The training process stops when generated data are not distinguishable from the real data.

3.2 GANs for Printed Image Simulation

Printed image simulation could be considered as an image-to-image translation problem. As shown in Figure 3.6, the original image is a digital image and the output is a simulated image. The GAN's model maps the digital image domain to the corresponding printed image domain. There are some typical applications in image-to-image translation based on generative models such as realistic samples for artwork [78], super-resolution [79] and colorization [80].

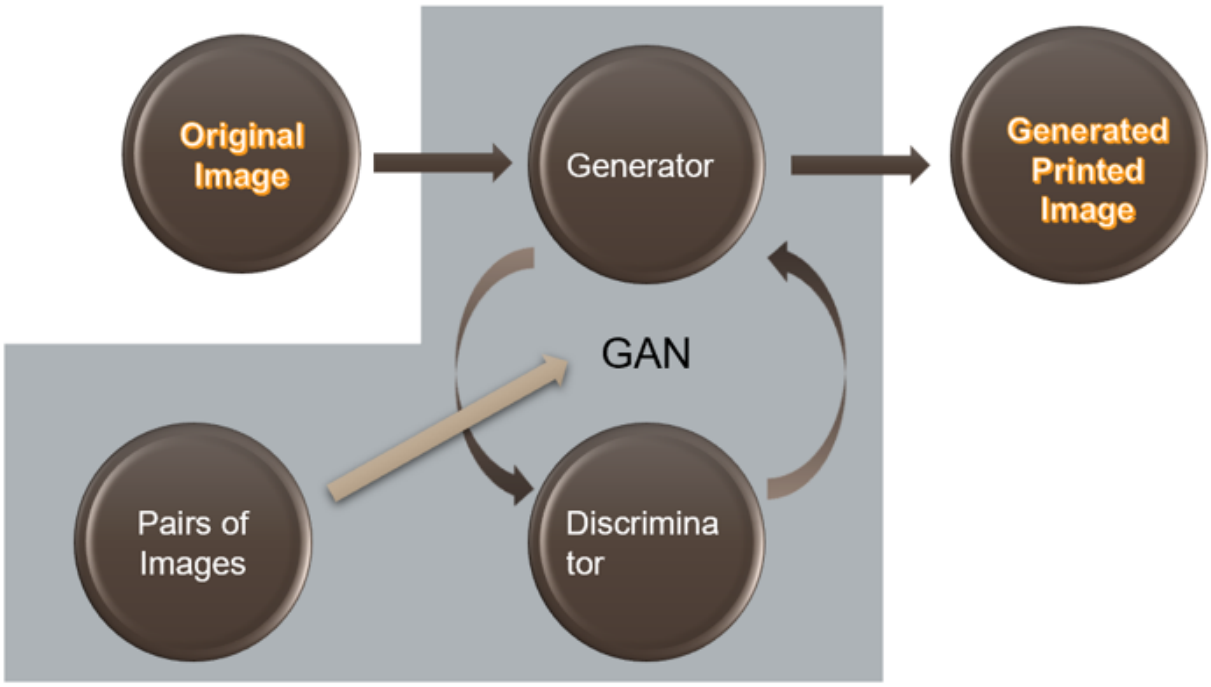
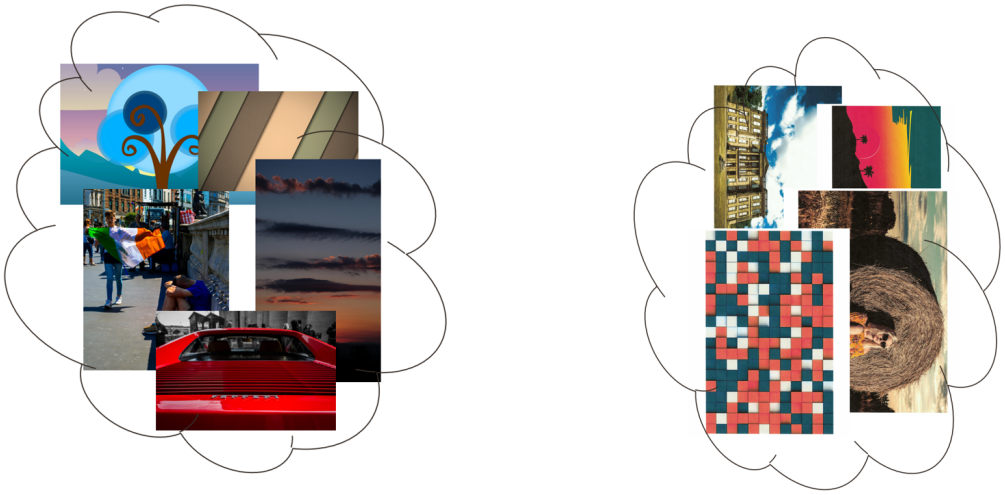


Figure 3.6. Image translation by GAN.

Generally, there are two approaches for solving the style translation problems: supervised learning and unsupervised learning. In the unsupervised learning scenario, a large number of images in two groups are separated by their style. However, in the training dataset, there are no pairs of images with a strong connection, as shown in Figure 3.7. Under certain architectural constraints between the two groups, the features could be learned by GANs via unsupervised learning. In another scenario called supervised learning, pairs of images are available in the training set, as shown in Figure 3.8. The input of the network has two related images belonging to two different groups.

3.2.1 Unsupervised Image-to-image Translation GANs

In this subsection, we use UNsupervised Image-to-image Translation networks [81] (UNIT) to solve the printed image simulation problem. Some of the printed images are collected without their corresponding digital reference images. Thus unsupervised learning is more applicable for this type of digital to printed image style transformation.



(a) Original image. (b) Printed images.

Figure 3.7. Two groups of images in unsupervised learning.

(a) Original image. (b) Printed images.

Figure 3.8. A pair of images in supervised learning.

Network Structure of UNIT GANs

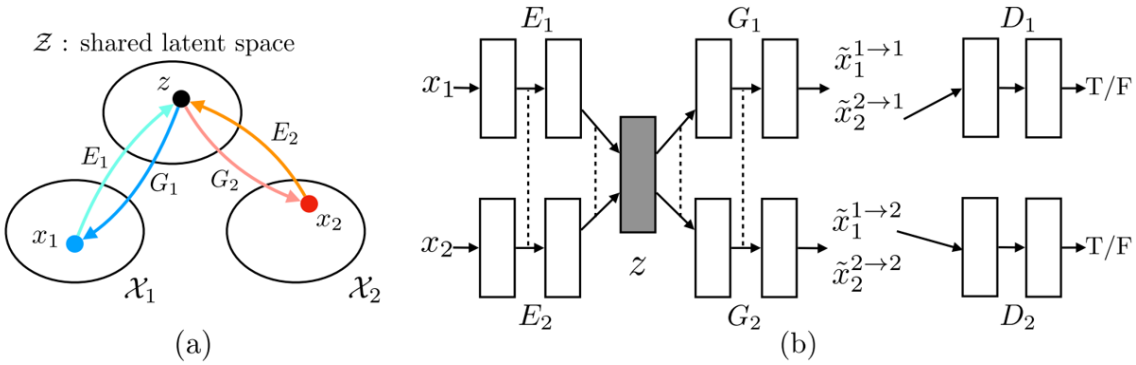


Figure 3.9. UNsupervised Image-to-image Translation networks [81].

Figure 3.9 is the structure of UNsupervised Image-to-image Translation networks. As shown in Figure 3.9 (a), the main assumption of this GANs is for image x_1 and image x_2 to be from two different domains χ_1 and χ_2, and to have a common shared code z in shared latent space Z. There are two mappings G_1 and G_2 which are generators from latent code z to image x_1 and x_2, respectively. In the opposite direction, there are two encoding functions E_1 and E_2 mapping x_1 and x_2, respectively to z.

In UNIT GANs, $\{E_1, G_1\}$ could be interpreted as a Variational Auto-Encoders sub-network and $\{E_1, G_2\}$ as an image-to-image translator sub-network. Respectively, $\{E_2, G_2\}$ could be interpreted as a VAEs sub-network and $\{E_2, G_1\}$ as an image-to-image translator sub-network. D_1 and D_2 are the adversarial discriminator sub-networks which classify the images as generated or real images. $\{G_1, D_1\}$ and $\{G_2, D_2\}$ could be treated as GAN sub-networks. E_1, E_2, G_1, G_2 are represented by four separated CNNs. Under the assumption of a shared latent space, the weights of last few high-level layers of E_1 and E_2 share the same value. Similarly, the weights of G_1 and G_2 are tied.

After the training process, image-to-image translation from domain χ_1 to domain χ_2 is done by an input image x_1 going through sub-network E_1 and G_2 to generate simulated image $\tilde{x}_1^{1\to2}$. In another direction, image-to-image translation from domain χ_2 to domain χ_1

is done by an input image x_2 going through sub-network E_2 and G_1 to generate simulated image $\tilde{x}_2^{2\to1}$.

Data Preparation and Training

The training set has 26 images in total. It is separated into two sets: the digital images set and the printed images with defects set. Each set has 13 images.

For printed image simulation, we increase the output resolution for the last layer from 256 to 512. However, the GPU has limited memory for the large GANs model. In order to fit the input and weights of the model to GPU memory, we modify the last 2 output layers from 32 to 16.

Learning Result for Group Mapping

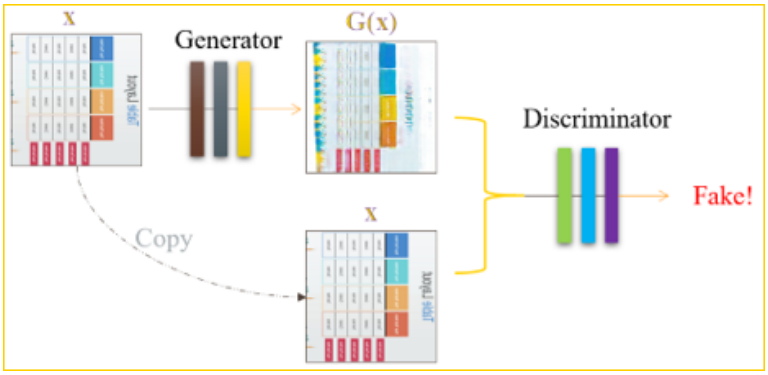
In the trial of group mapping simulation, the image becomes lighter after simulation after 86000 iterations, which resembles some printed images having a lighter color. A pre-processing step such as registration for data, is not required for the training dataset. However, the test result is not satisfactory since it is far lighter than the desired ground truth. The training dataset is too small for training a large UNIT GANs model. Essentially, the unsupervised image-to-image translation problem is considered more difficult than supervised translation. Thus, in the following subsection, we consider the digital to printed image translation problem as a supervised 1-to-1 image mapping problem.

3.2.2 Supervised Image-to-Image Translation GANs

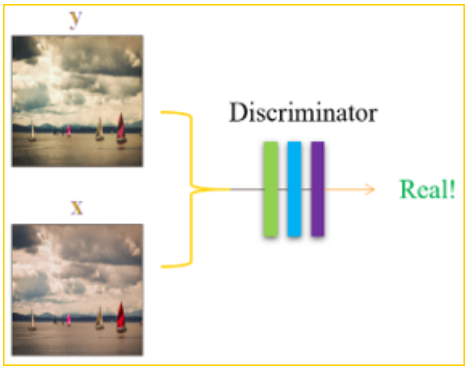
PIX2PIX GANs

In this section, we propose a deep learning framework based on pix2pix (pixel to pixel Conditional Adversarial Networks) [82] GANs for simulating printed image datasets, which is used for training printed image quality assessment or defects detection algorithms. In contrast to unsupervised mapping for image groups, the supervised image-to-image mapping provides better image quality under strong supervised constraints.

Most traditional supervised image-to-image translation approaches minimize pixel-wise Euclidean distance between the output and ground truth images. Some researches [83], [84] have addressed the fact that the L2 loss function tends to generate blurry images. When traditional approaches are trying to minimize the Euclidean distance loss function, the outputs are averaged to decrease the loss, which is the main reason for blurry outputs. Under the assumption of GANs, the discriminator tends to classify the blurry images as fake and forces the generator to reproduce clear images.



(a) Generator and Discriminator.



(b) Discriminator.

Figure 3.10. Pix2pix GAN structure.

In the pix2pix translation, the networks are composed of two sub-networks named generator and discriminator. The generator is a network randomly generating some images from the original images under certain constraints, such as the L1 constraint, to create new images with a different style. The discriminator is another network to distinguish generated images from real images. A mapping between input and output images could be learned by training on the aligned image pairs.

As shown in Figure 3.10a, when training the conditional GANs mapping images from digital to printed, the discriminator D is trained to classify between fake (simulated by the generator) and real (scanned printed image) images. The generator G is trained to confuse the discriminator D. The main difference between pix2pix conditional GANs and unconditional GANs is the input for generator G and discriminator D. The input is not a simulated printed image or real scanned printed image, but a 2-tuple consisting of a printed image and a digital image. As shown in Figure 3.10a, the input to D is a tuple $(G(x), x)$, where $G(x)$ is the generated printed image (fake) and x the original digital image. As shown in Figure 3.10b, the input to D is a 2-tuple (y, x), where y is the scanned printed image (real) and x the original digital image.

Experiment on Digital to Monocolor Printed Images Translation

In this part, we implement image-to-image translation from digital to monocolor printed image without defects by pix2pix GANs. The original images are with a resolution of 2624×1946. There are 13 pairs of images that are the same as the dataset in Subsection 3.2.1. In order to increase the number of training images and fit the weights of the model to the GPU memory, the original images are cropped into images with a resolution of 656×486. Among the cropped images, 144 images without defects are selected for this part. There are 104 pairs of images as the training set and 40 as the test set. After that, two related images are combined into an image pair as the input of the pix2pix GANs, as shown in Figure 3.11.

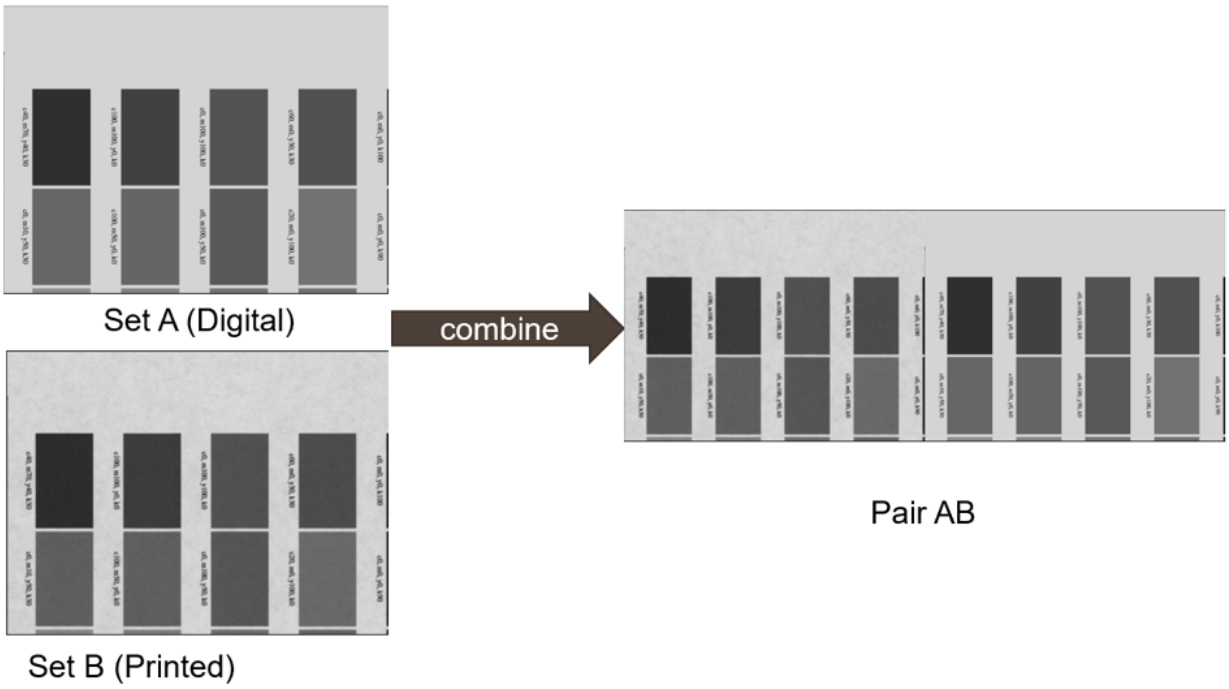


Figure 3.11. Data preparation: two cropped images are combined into an image pair as the input of the pix2pix GANs

After 6000 epochs of training, the testing results, as shown in Figure 3.12, 3.13, 3.14 and 3.15, prove the pix2pix GANs under conditional constraint could be trained for digital to printed image translation.

Specifically, in Figure 3.12 and 3.13, the simulated images have small mottles that resemble the printed image on the right side of the image. Figure 3.14 is a failure case of our pix2pix models for printed image simulation, which might be caused by mode collapse described in [85]. The printed image generation GANs model is trained only with a dataset of 104 pairs of images. Figure 3.15 is another example of simulated images. Although the simulated image is different from the real printed image on the right side, from Figure 3.15 we could see the GANs model learns the different possible modes generated in the printing process.

20180620_152942_icf_compare_Any_Green_152.PSA.VSLeft_0_3

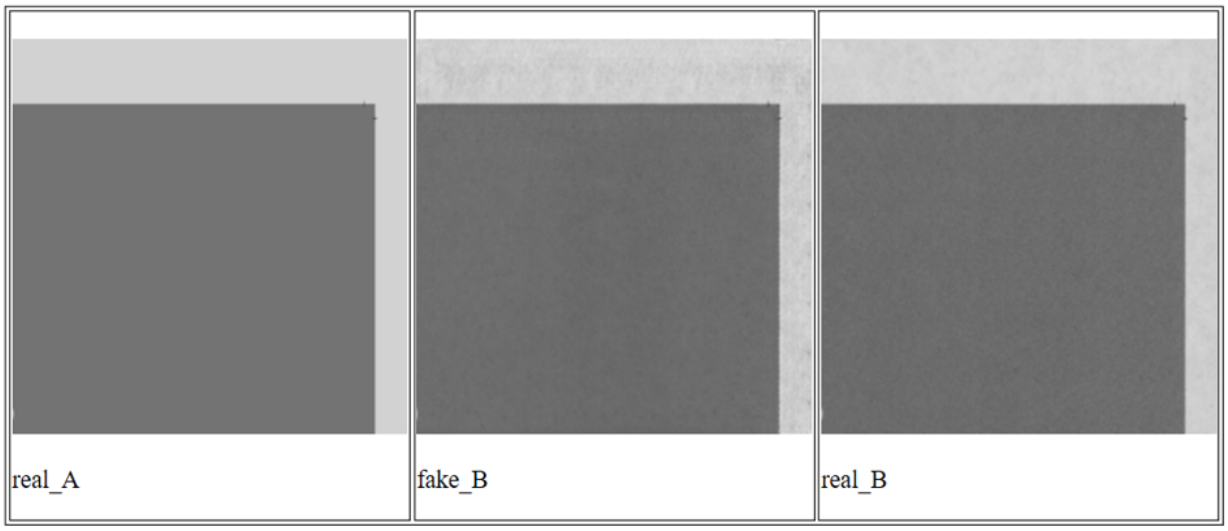


Figure 3.12. Example 1 of the test image, the left is the digital image, the middle is the simulated image, the right one is the real printed image

20180620_152942_icf_compare_Any_Green_152.PSA.VSLeft_3_0

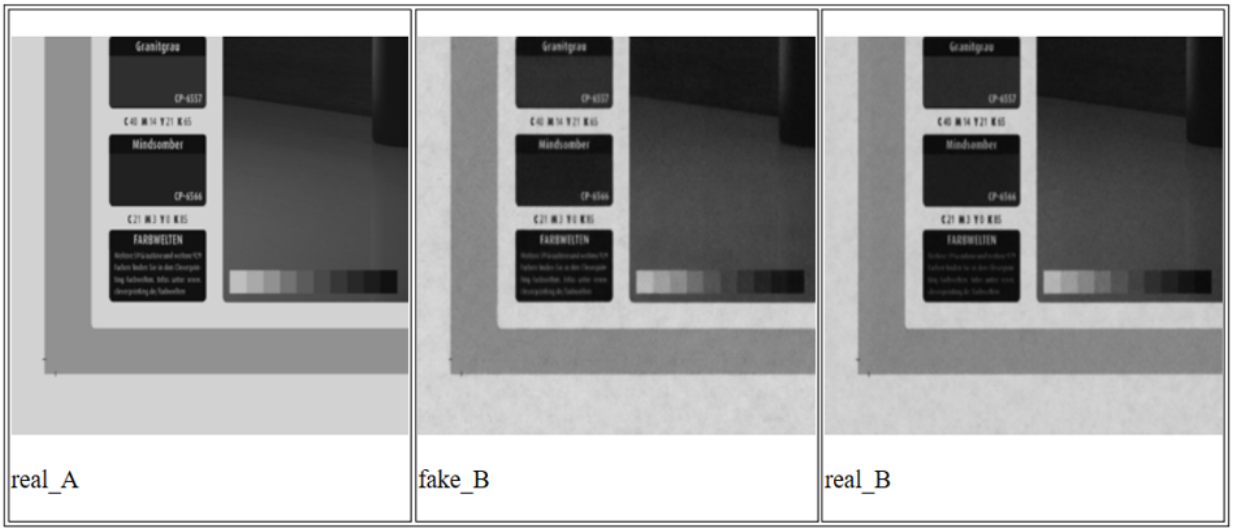


Figure 3.13. Example 2 of the test image, the left is the digital image, the middle is the simulated image, the right one is the real printed image

20180620_152942_icf_compare_Any_Green_152.PSA.VSLeft_1_2

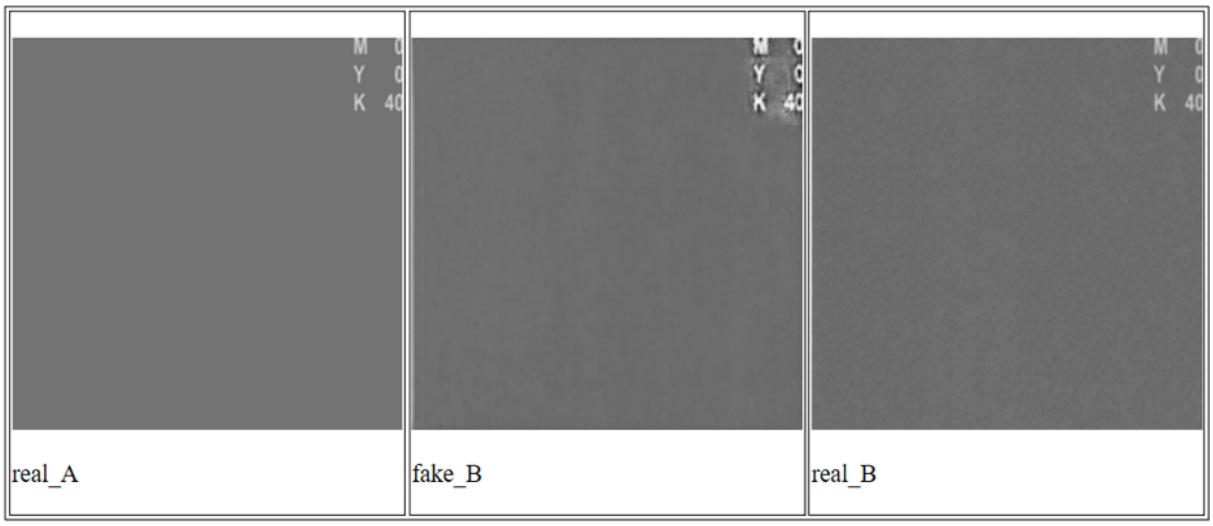


Figure 3.14. Example 3 of the test image in a failure case, the left is the digital image, the middle is the simulated image, the right one is the real printed image

20180620_152942_icf_compare_Any_Green_164.PSA.VSLeft_0_1

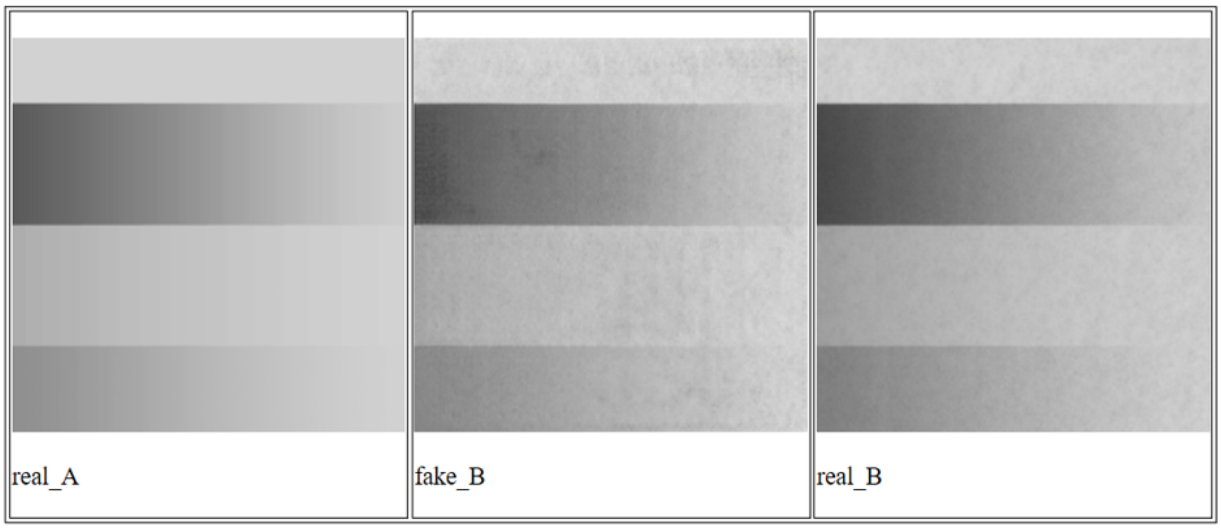
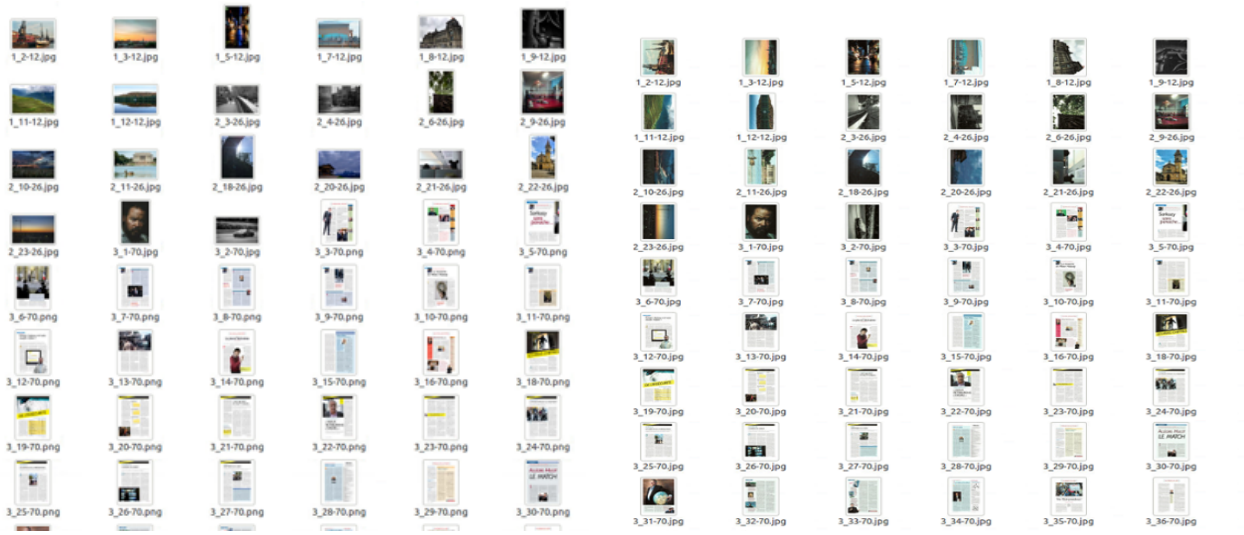


Figure 3.15. Example 4 of the test image, the left is the digital image, the middle is the simulated image, the right one is the real printed image

Experiment on Digital to RGB Printed Image Translation

In this part, we implement image translation from digital to printed image (without defects) for RGB images by pix2pix GANs. The digital to printed image translation aims to simulate the printing process without printing and scanning the image. A dataset with 96 pairs of images is collected, including realistic images (camera photos), synthetic images (from computer games simulation), and documents. Those pairs of images are printed and scanned by an HP MFP586 jet-ink printer, as shown in Figure 3.16.



(a) Digital. (b) Printed.

Figure 3.16. Digital to printed RGB images dataset.

For the images directly collected from the scanner as shown in Figure 3.16b, they are not well aligned and have superfluous blank areas. After implementing the image registration algorithm, the aligned image pairs, as shown in Figure 3.17, are used for training and testing. The image registration approaches are introduced in Chapter 4.

(a) Digital. (b) Printed.

Figure 3.17. Example of an image pair.

In this work, the pix2pix GANs are trained with 74 pairs of registered digital to printed RGB images, as shown in Figure 3.18.

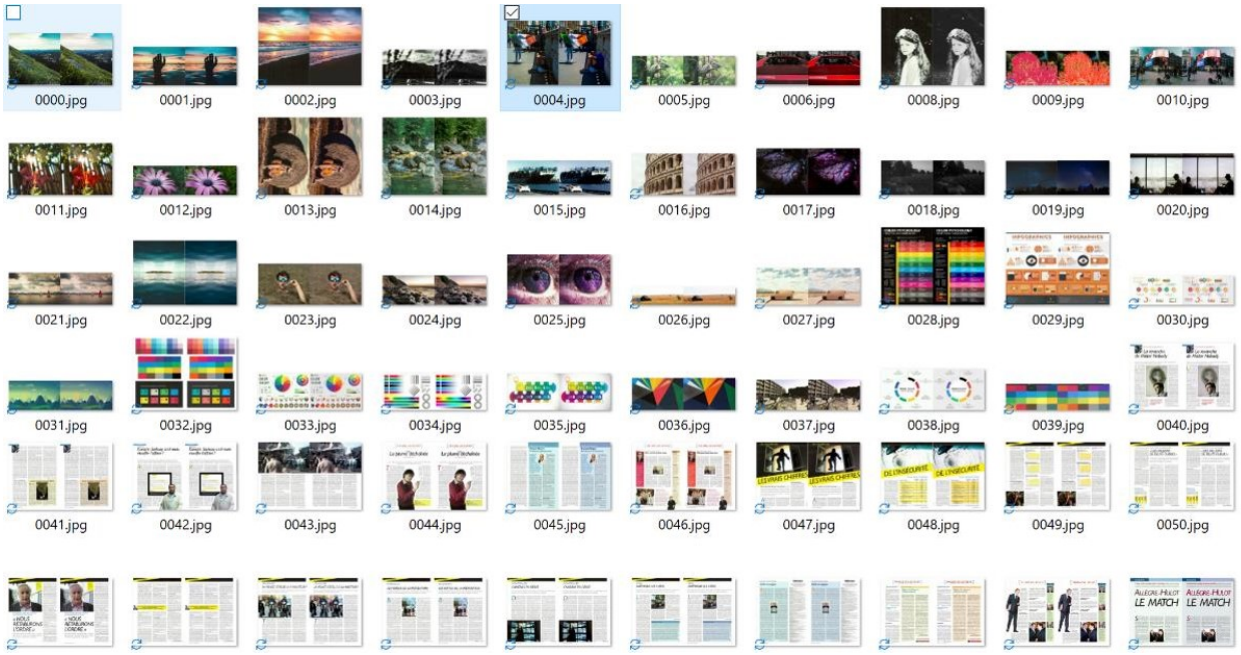


Figure 3.18. Digital to printed RGB images training set with 74 pairs of images.

The test set consists of 22 pairs of registered digital to printed RGB images, as shown in Figure 3.19.

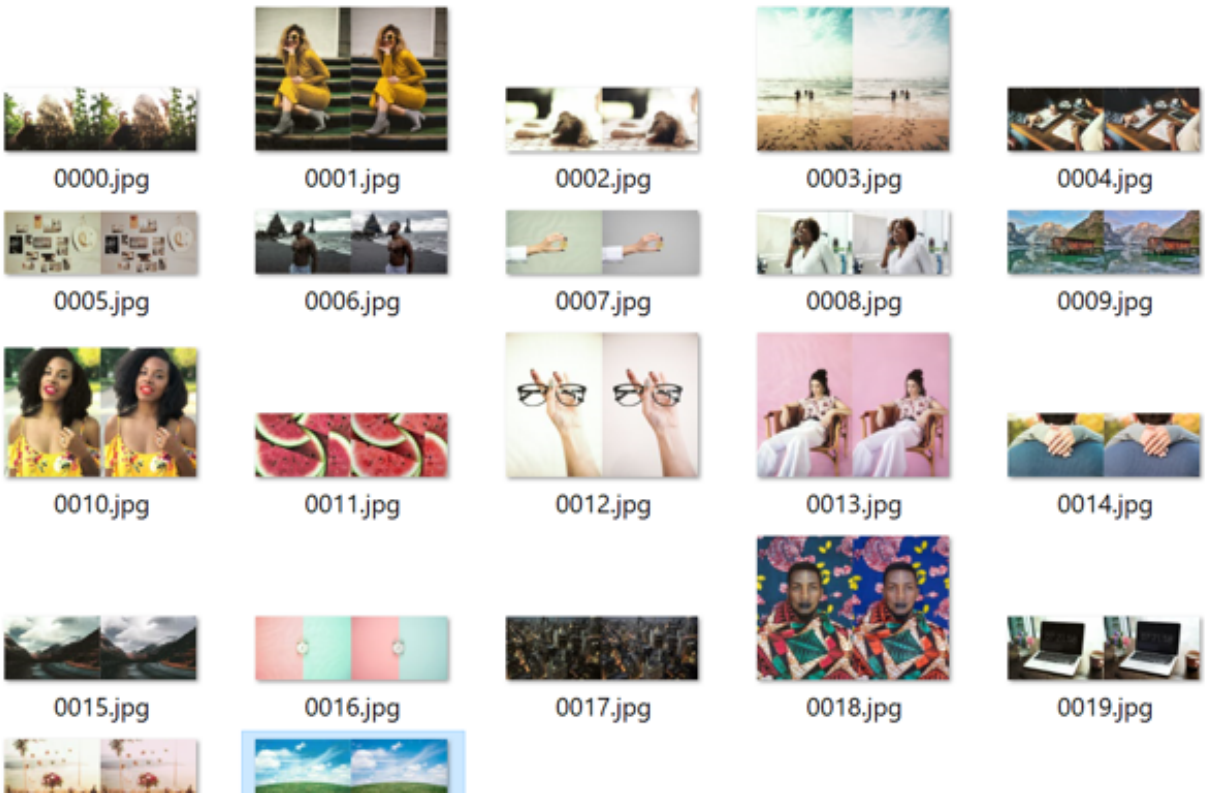


Figure 3.19. Digital to printed RGB images testing set with 22 pairs of images.

In the inference stage, digital images from the testing set are input to the trained generator network to provide printed image simulations. For the training result after 6000 epochs, Figure 3.20 and 3.21 are examples of pix2pix GANs simulation for RGB printed image. A mapping between a digital image and a printed image fits the image-to-image translation paradigm.

Figure 3.20. Example 1 of output for test images, the left is the digital image, the middle is the simulated image, the right one is the real printed image.

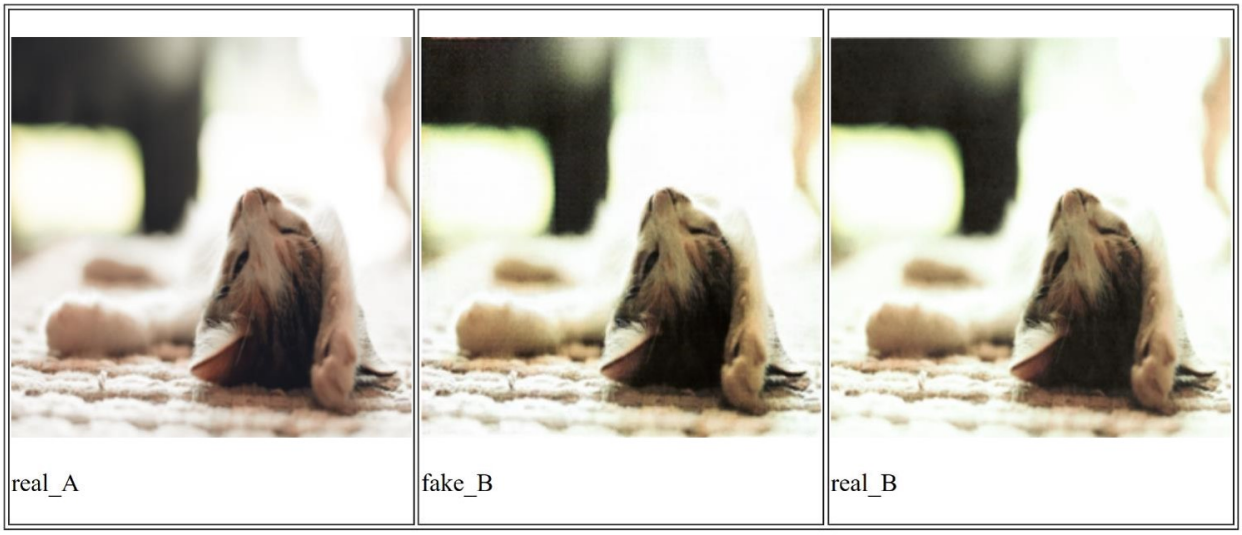


Figure 3.21. Example 2 of output for test images, the left is the digital image, the middle is the simulated image, the right one is the real printed image.

The qualitative results show that color and texture could also be learned in our simulation. As shown in Figure 3.22 and 3.23, the color has been changed for the simulated printed image, which resembles the process of printing.

Figure 3.22. Example 3 of output for test images, the left is the digital image, the middle is the simulated image, the right one is the real printed image. The color changes in the simulated image match the real printed image compared to the original digital image.

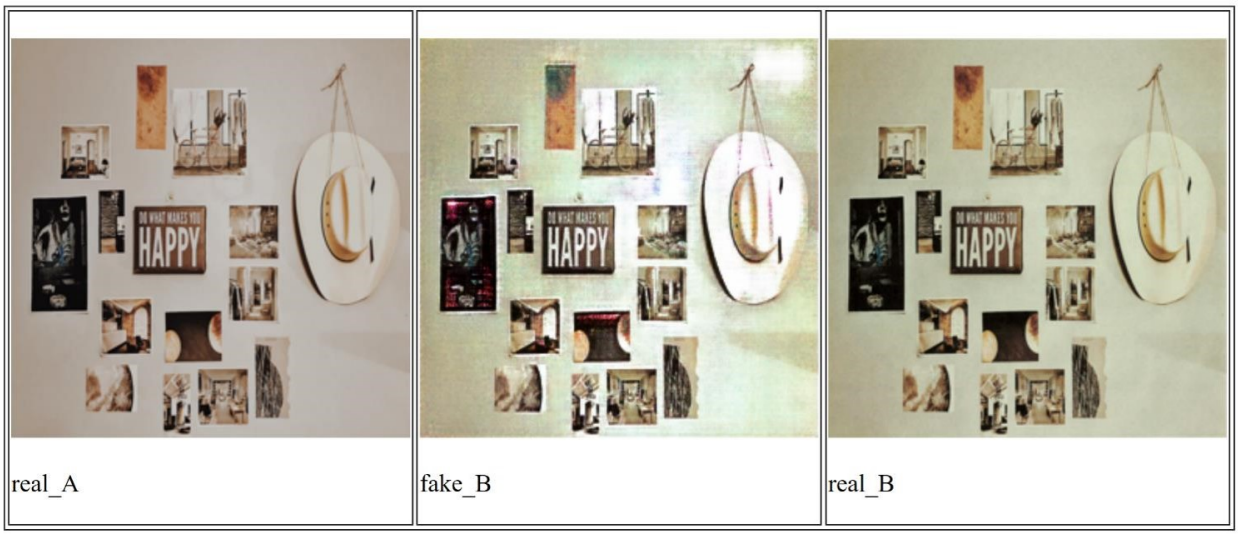


Figure 3.23. Example 4 of output for test images, the left is the digital image, the middle is the simulated image, the right one is the real printed image. The simulated image is a failure case since it introduces extra artifacts into the original image.

As shown in Figure 3.24, the images are zoomed in for the texture change to be observed in the simulated image compared to the original digital image.

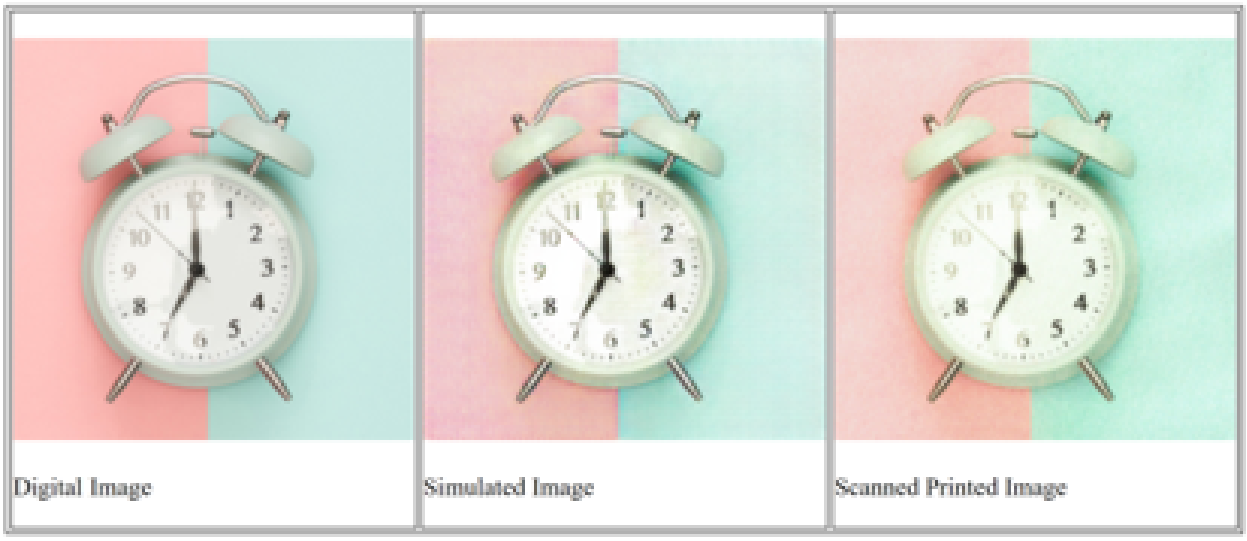


Figure 3.24. Example 5 of output for test images, the left is the digital image, the middle is the simulated image, the right one is the real printed image. The images are zoomed in. Thus, the texture change in the simulated image is visible compared to the original digital image

3.3 Conclusion

For deep learning-based defect classification and detection algorithms, numerous printed and scanned images are required. Compared to other methods using actual printed images as training data, the photorealistic printed images are automatically generated in our proposed framework. Thus, we can reduce the printing and scanning time and cost in the data collection stage.

The Generative Adversarial Network is an efficient tool for generating simulation data for training classification and detection algorithms. Thus, a deep CNN can be trained with a limited amount of scanned and digital image pairs. The qualitative results show that we can reduce the effort to print and scan many defective images since the GAN data augmentation could ideally produce abundant data. The GANs simulation approach leads to a simulated dataset with more high-quality images.

4. PRINTED IMAGE REGISTRATION

4.1 Introduction

Image registration is a spatial transformation that maps points from one image to corresponding points in another image: matching two images so that corresponding coordinate points in the two images are for the same physical region of the scene being imaged, as shown in Figure 4.1. The image registration problem is also named image fusion, superimposition, matching, alignment or merge.

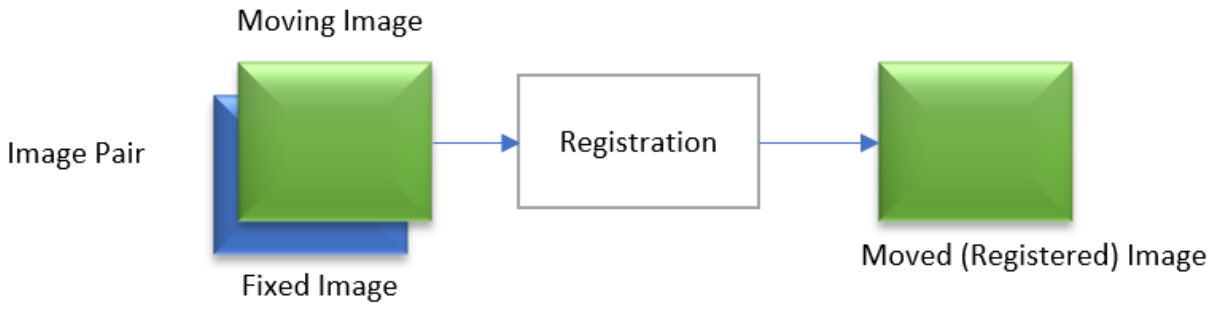


Figure 4.1. Example of the image registration process.

In this chapter, we focus on the printed image registration problem. As shown in Figure 4.2, the input of the registration has two monochrome or colored images in a pair. The registration algorithm is to align the moving image with the fixed image. The output is the registered image after spatial deformation, which should match the fixed image. We also investigate medical 2D images to verify the generalization of our proposed model.

4.1.1 Applications

Image registration is an essential pre-processing step for printed image quality assessment and defect detection. The profile of printed defects can be extracted from the difference between the printed and digital images if two images in a pair are accurately aligned. For example, in [86], Xiao proposed a method by matching feature pairs for print quality diagnostics, as shown in Figure 4.3.

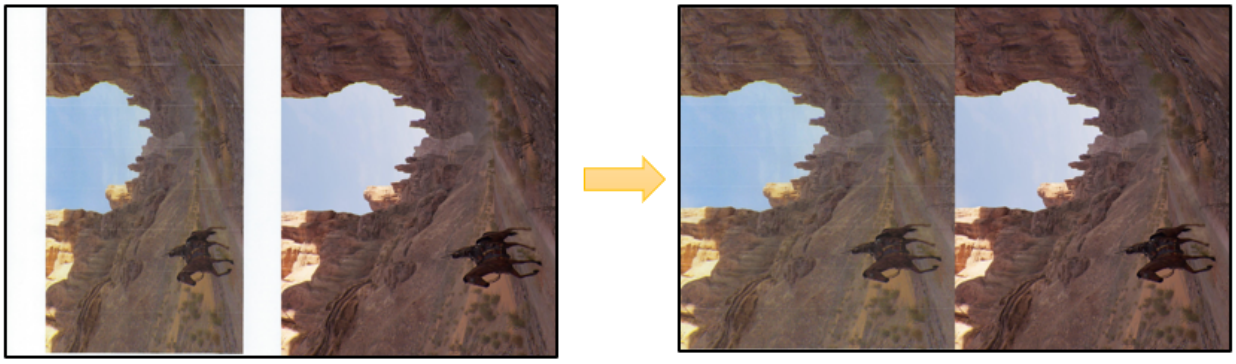


Figure 4.2. Example of the printed image registration process: the left is the image pair before registration, the right is the image pair after registration.

Figure 4.3. Example of the printed image registration by the pair matching method for image quality diagnostics [86].

Besides printed image quality assessment, image registration is widely used in computer vision, video analysis, medical image processing, material mechanics, and remote sensing.

4.1.2 Problem Formation

Image registration methods can be divided into several categories according to different attributes. By the nature of their basis, there are feature-based methods and intensity-based methods. Feature-based methods find correspondence between image features, such as points, lines and contours, and then match the features. Intensity-based methods compare intensity patterns in images via similarity metrics, such as RGB values, light intensity values, or grayscale values.

By the transformation domain, image registration methods can be divided into global and local methods. Global methods are usually rigid registration. For rigid image registration, the number of parameters depends on the type of transformation, such as rigid, affine and projective. Local methods require deformable registration. Deformable registration is a pixel-wise mapping, which requires more parameters proportional to the input image resolution.

The registration parameters update can be decided by an optimization procedure such as Gradient Descent, RMSprop, Adam, AdaMax [65], etc. The modalities of inputs involved in registration include mono-modal, multi-modal and modality to model [87], [88]. The dimensionality of the input can be classified as 2D-2D, 3D-3D and 2D-3D [89].

The input images are in the D-dimensional coordinate space ϕ. For the printed image registration problem, we mainly focus on images in 2D space. The registration inputs I_{fixed} and I_{moving} are defined as the fixed image and moving image, respectively. I_{fixed} and I_{moving} are functions $\phi \longrightarrow \mathbb{R}^c$, where c is the number of channels. For a monochrome image, $c = 1$; for a color image, $c = 3$. The deformation field is defined as a mapping $\phi : \phi \longrightarrow \phi$ between two image coordinate spaces. The objective of image registration is to predict a deformation field ϕ from input I_{fixed} and I_{moving}. The deformation field ϕ warps the moving image I_{moving} to a warped image I_{moved}: $I_{moved} = \phi \circ I_{moving}$.

4.2 Global Image Registration

In image registration, global rigid image registration and local deformable image registration are processed independently. Global image registration means using a rigid global transformation such as similarity, affine or projective transformation to align two images. At first, the image is globally registered and then locally aligned to get a better visual result.

4.2.1 Feature-based Image Registration

For global image registration, RANSAC [90] is a traditional method based on the extracted image features. As shown in Figure 4.4, feature-based methods extract features such as ORB [91] and GeoDesc [92], and find correspondence between image features by RANSAC to estimate the global transformation parameters. An extra affine tuning step is used in the printed image registration pipeline by the least squares method.

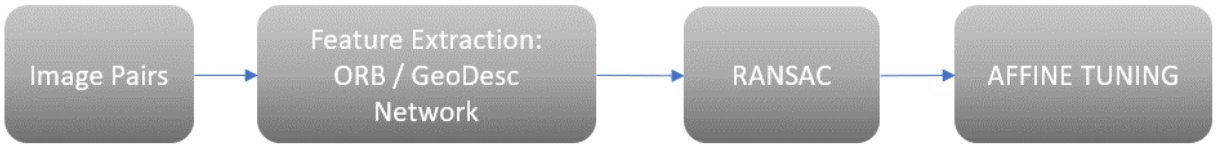


Figure 4.4. Image registration pipeline for printed image quality assessment.

Figure 4.5 shows the matching result of a deep learning-based feature extractor. Figure 4.6 shows the registration result by a feature-based registration pipeline.

However, feature-based image registration methods fail in cases such as textureless images and repeated patterns, as shown in Figure 4.7. This method requires a robust image descriptor. However, the printed images often have different text content with repeated shapes which causes the failure of feature-based methods. Figure 4.8 is an example of feature matching by the descriptor. The left image is the moving image and the right image is the fixed image. The top row is the matching result by the SIFT descriptor [93]. The bottom row is the result of the GeoDesc descriptor. The mismatching pairs are dashed line circles with the same color. Both results show many mismatching pairs, which causes the failure of printed image registration.

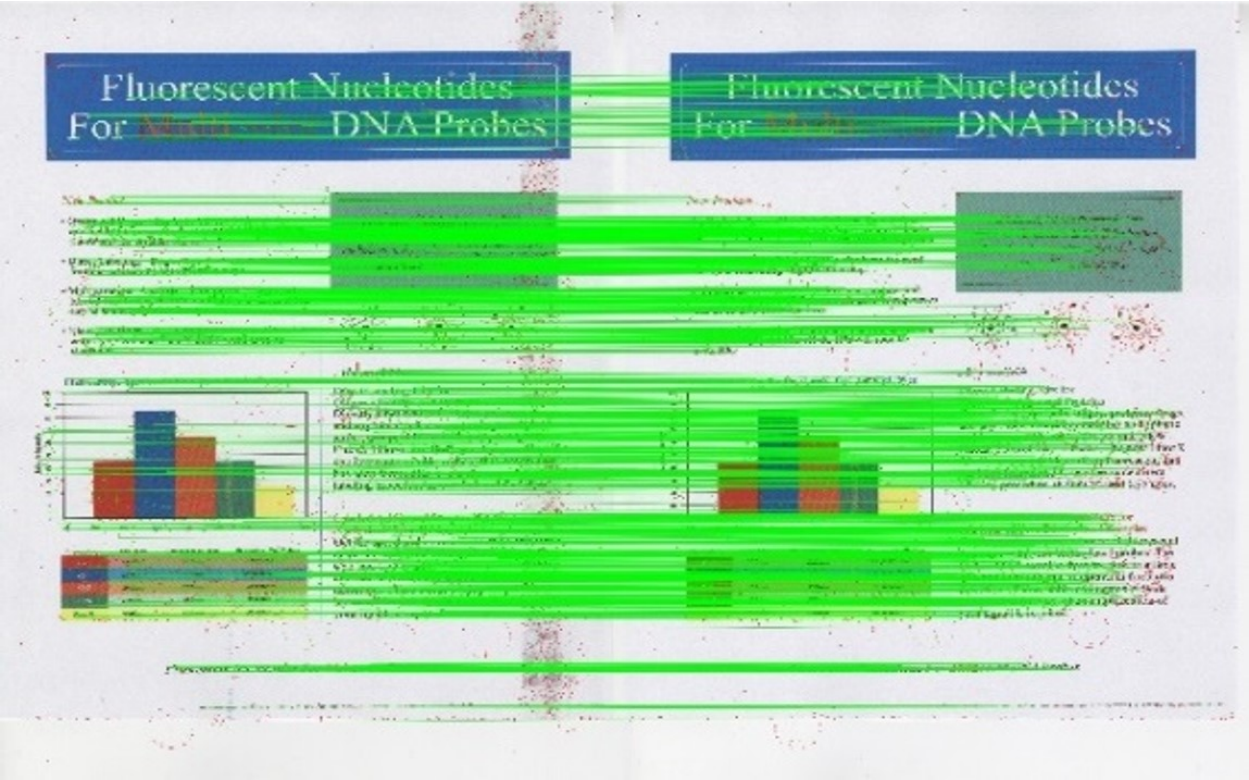


Figure 4.5. Example of the deep learning-based GeoDesc descriptor image matching .

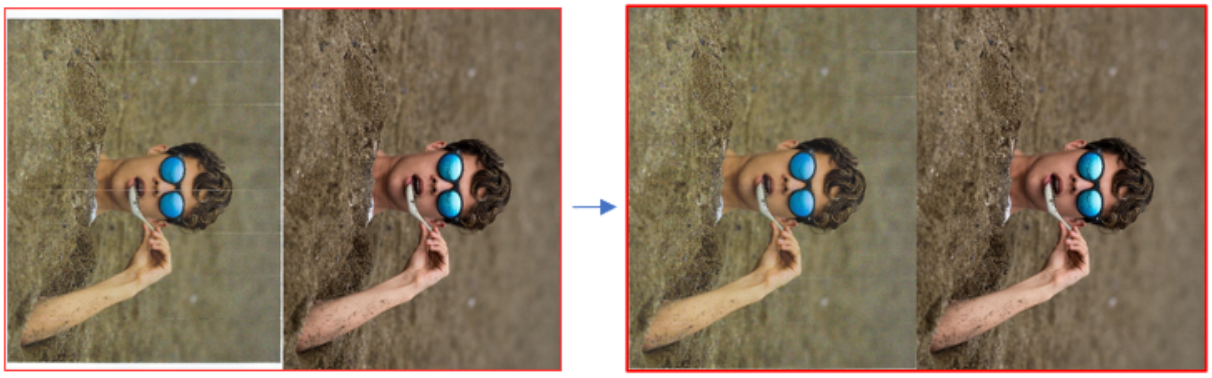


Figure 4.6. Example of an image registration result by the proposed pipeline.

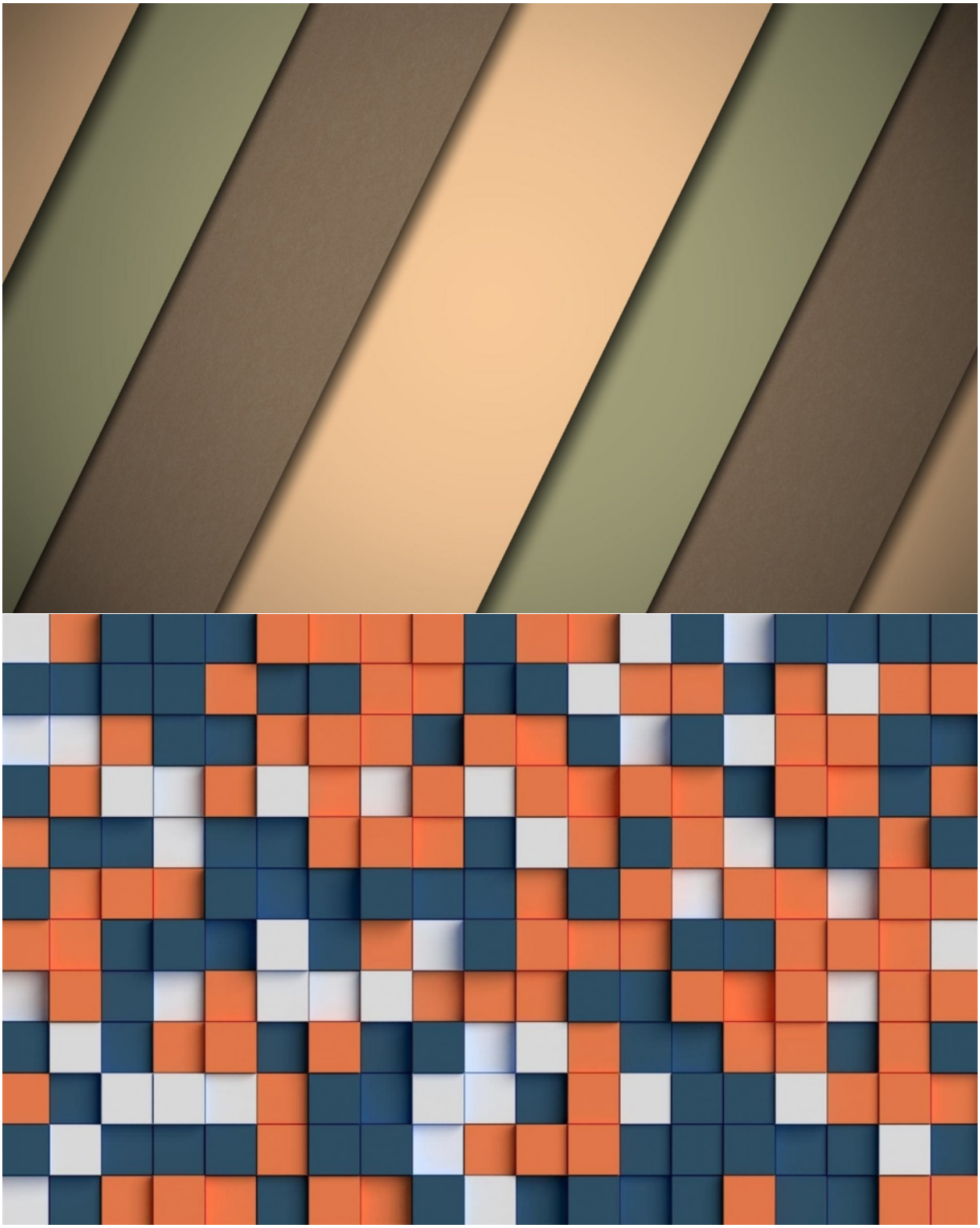


Figure 4.7. Failure cases of the feature-based method.

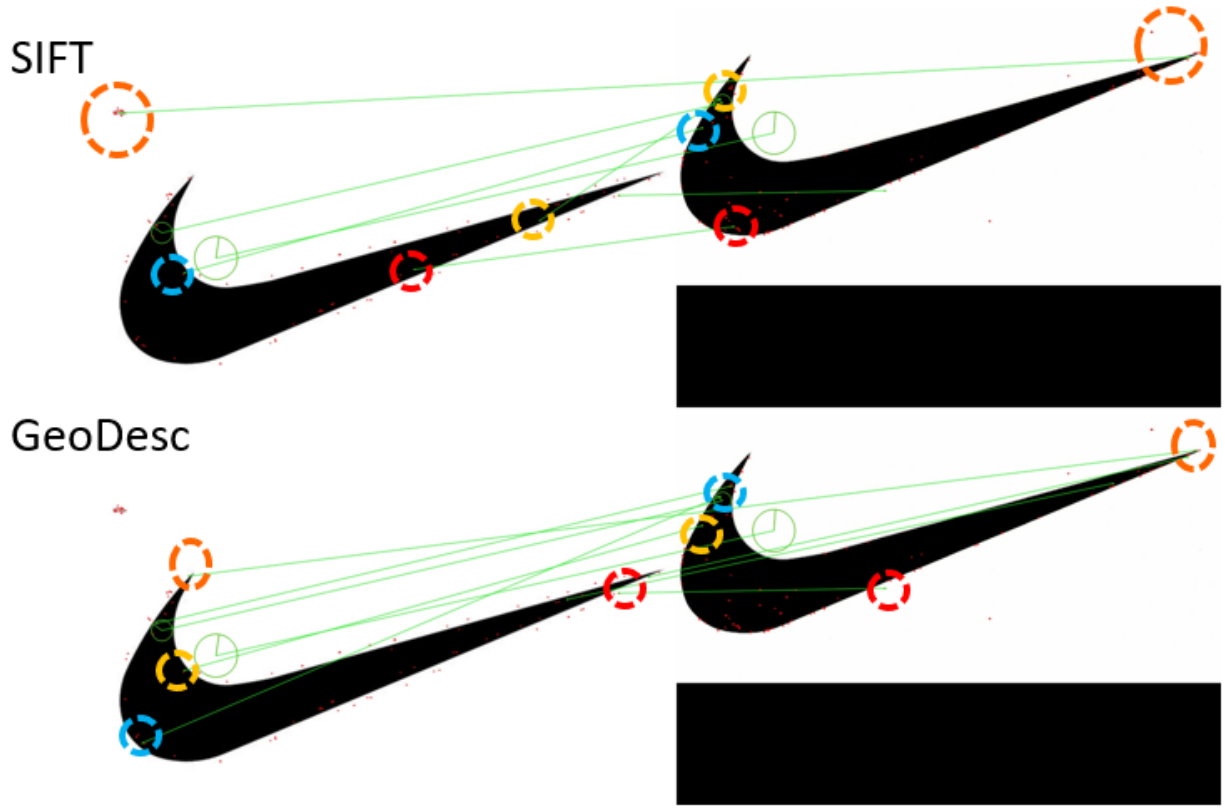


Figure 4.8. Failure cases of the feature-based method. The circles with the same color indicate the mismatching pairs.

4.2.2 Intensity-based Image Registration Algorithm using Deep Learning Framework PyTorch

Another class of global image registration methods is the intensity-based method which compares intensity patterns between images via similarity metrics such as the sum of squared differences, normalized cross correlation and mutual information. Intensity image registration iteratively computes the similarity loss and updates the transformation parameters using an optimizer.

Thanks to the immense growth in the deep learning society in recent years, there are many open-source deep learning software frameworks such as TensorFlow [94], Pytorch [95], [96] and Caffe [97] which have the advantage of parallelism and auto-gradient computation. In this work, we propose an intensity-based printed image registration method using the deep learning framework Pytorch. The backward gradient computation in image registration can

be achieved by the automatic differentiation module in the framework. The affine transformation matrix multiplication is similar to the tensor multiplication operator in Pytorch. The warping of the deformation field can leverage the spatial transformer networks. Moreover, the grid sampling module can be used for bilinear interpolation. The parallel computation of the GPU can speed up all of the matrix and vector operations.

As shown in Figure 4.9, we initialize the transformation parameters P and translate P to transformation matrix T. Usually, the transformation matrix T is an identity matrix at first, which means there is no shift for the corresponding pixel coordinates in the early stage. The transformation matrix T is converted to the deformation field ϕ, which has the shift information for each coordinate. The moving image I_{moving} is then warped by the deformation field ϕ and is transformed to moved image I_{moved}. The similarity loss L_{sim} is computed by comparing the difference of fixed image I_{fixed} and moved image I_{moved}. The similarity loss L_{sim} is used to compute the gradient $\frac{\partial L_{sim}}{\partial P}$ relative to the transformation parameters. The parameter difference ΔP is computed by combining the gradient $\frac{\partial L_{sim}}{\partial P}$ with the learning rate λ. The parameters P are then updated by ΔP. The similarity loss L_{sim} is converged iteratively by repeating the above process. When $L_{sim} < \epsilon$ or the iteration number i exceeds a certain threshold n, we have the transformation parameters P, the fixed image I_{fixed} and moved image I_{moved} as the final registration result.

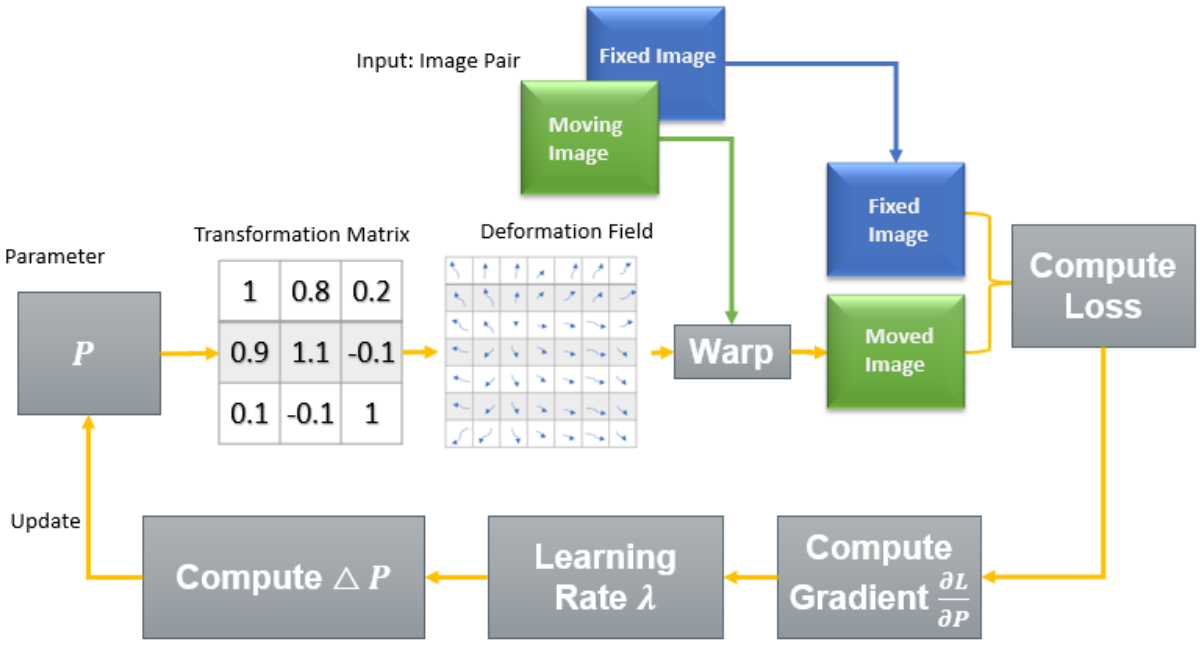


Figure 4.9. Intensity-based registration (global) using deep learning framework PyTorch.

The Spatial Transformer Network (STN) [98] is a network operator used to transform spatial tensor features to warped features. A deep learning framework such as TensorFlow or PyTorch uses the differential STN module to map features between two CNN layers and perform bilinear interpolation at the same time. Figure 4.10 shows the process of computing a pixel value by bilinear interpolation. The pixel value is defined as a^s_{ij}, where i, j are the indices of the coordinate, s means the layer after registration and $s-1$ means the layer before registration. In STN, $s-1$ is the CNN feature layer before STN operator and s is the feature layer after STN operator. For example, in Figure 4.10, we want to compute the pixel value a^s_{11} after registration. The affine transformation matrix T can be represented as a rotation matrix $R = \begin{pmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{pmatrix}$ and shift (translation) vector $V = \begin{pmatrix} 0.6 \\ -1 \end{pmatrix}$. So the indices of the coordinate in the previous $s-1$ layer (before registration) is computed as:

$$\begin{pmatrix} i' \\ j' \end{pmatrix} = R \times \begin{pmatrix} i \\ j \end{pmatrix} + V = \begin{pmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0.6 \\ -1 \end{pmatrix} = \begin{pmatrix} 0.6 \\ 0.414 \end{pmatrix}$$

Ideally, the pixel value $a^{s-1}_{0.6,0.414}$ of the previous $s-1$ layer is at coordinate $(0.6, 0.414)$. However, we cannot directly get the pixel value of the $s-1$ layer at the floating point coordinate indices. So the bilinear interpolation is used for the calculation of the pixel value:

$$\begin{aligned} a^{s}_{11} =& a^{s-1}_{0.6,0.414} \\ =& (1-0.6)\times(1-0.414)\times a^{s-1}_{00} \\ =& (1-0.6)\times(1-0.586)\times a^{s-1}_{01} \\ =& (1-0.4)\times(1-0.414)\times a^{s-1}_{10} \\ =& (1-0.4)\times(1-0.586)\times a^{s-1}_{11} \end{aligned}$$

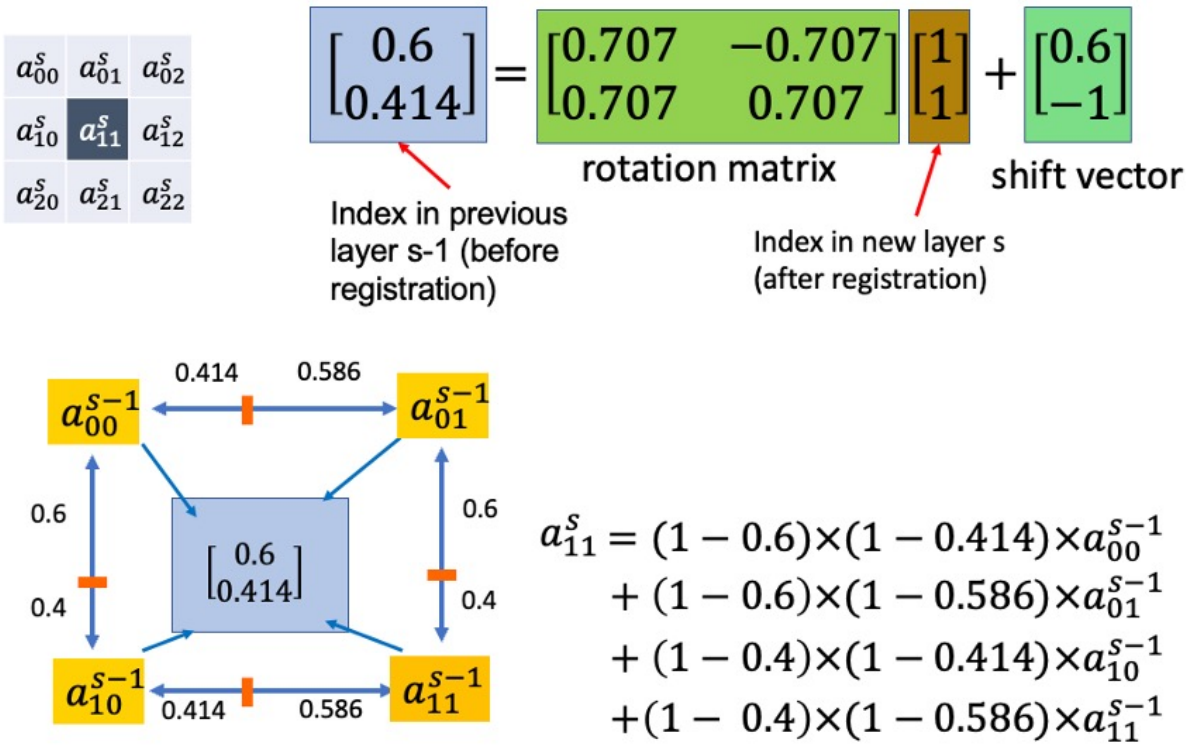
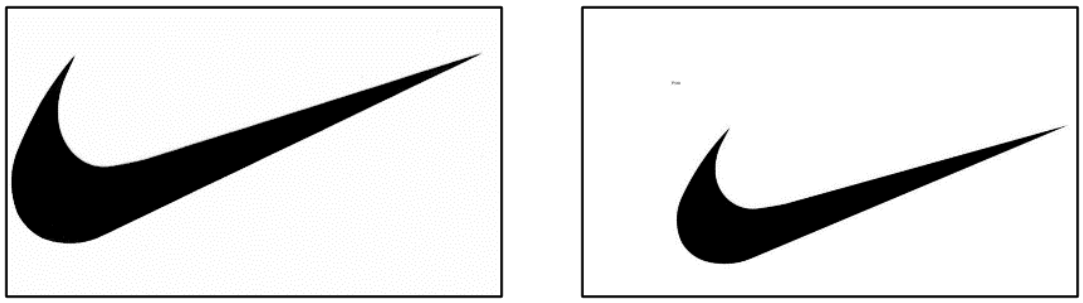


Figure 4.10. An example of the calculation of a target pixel value by bilinear interpolation.

In the experiment, the transformation consists of 4 degrees of freedom, including translation $(\Delta x, \Delta y)$ and scaling (l_x, l_y). The optimizer is Adam [99] with a learning rate of 0.01. The Mean Squared Error (MSE) is used as the similarity loss function L_{sim}. The inputs of

the registration experiment are shown in Figure 4.11a as the fixed image and Figure 4.11b as the moving image.



(a) Fixed image. (b) Moving image.

Figure 4.11. Experiment inputs of the intensity-base global image registration.

The experiment is initialized with the identity transformation, where $\Delta x = 0$, $\Delta y = 0$, $l_x = 1$ and $l_y = 1$. After an iterative intensity-based image registration process, the experiment result in Figure 4.12 is a merged image, which is a superimposed image of the fixed image and the moved image after registration. The magenta area in the merged image is the region with pixel values that differ from those of the fixed image.

Figure 4.12. The merged image is the experimental result of the intensity-based global image registration.

As shown in Figure 4.13, The similarity loss L_{sim} converges after around 180 epochs.

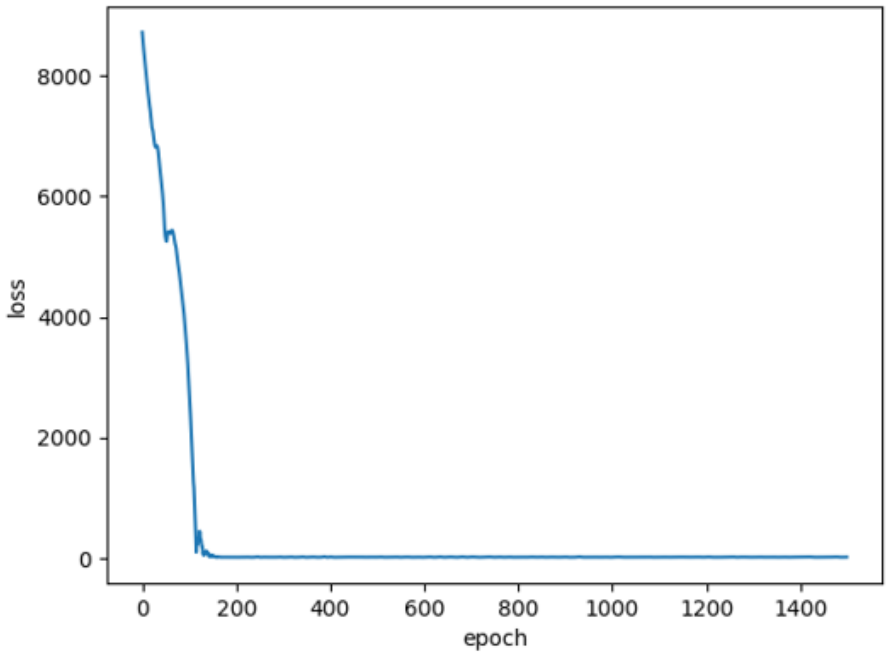


Figure 4.13. Experimental result: MSE loss plot across epochs.

4.3 Deformable Image Registration

Even with good rigid global image registration, local deformable misalignment still exists in cases such as printed images and medical images, as shown in Figure 4.14. In this example, the merged image after rigid global registration still shows some local misalignment for characters and lines. The local misalignment is because most image registration problems in real applications cannot be simplified to a global registration problem with limited parameters. For example, a Euclidean transformation has only 3 DOFs (Degrees of Freedom), an affine transformation has 6 DOFs, and a homography transformation has 8 DOFs. For each corresponding pixel pair between two images at coordinate (i, j), a local deformable transformation needs 2 DOFs $(\Delta x_{\mathrm{i,j}}, \Delta y_{\mathrm{i,j}})$. For pixel pairs in the entire image, the local deformable transformation needs $2 \times w \times h$ DOFs, where w is the width and h is the height of the image. Because deformations are similar for most of the neighborhood areas, we need to find a method to represent the shift for each pixel and keep the local similarity simultaneously. In the following subsections, we will introduce the U-Net architecture-based method

for printed images in Subsection 4.3.1, and propose a new recurrent network-based method named R-RegNet in Subsection 4.3.2. Subsection 4.3.3 introduces the unsupervised loss function used in the training process. In Subsection 4.3.4 and 4.3.5, we introduce the datasets and experimental results comparing different deformable image registration methods.

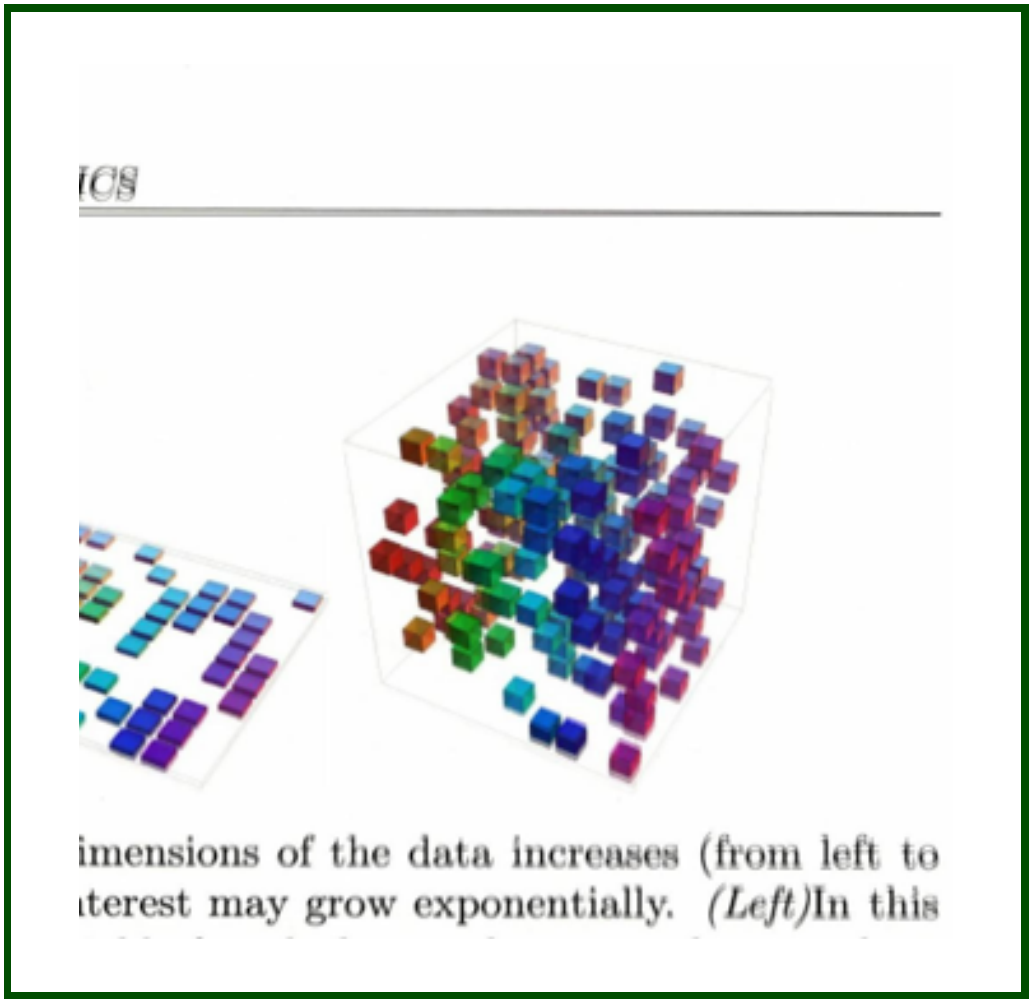


Figure 4.14. Example of local misalignment after global image registration.

4.3.1 U-Net VoxelMorph-based Method

For local medical deformable image registration, methods such as [100] and [101] use deep convolutional neural networks to register two images. The work in [102] introduces a network named Volume Tweening Network (VTN) for unsupervised 3D medical image registration

problems. A recent work named Recursive Cascaded Networks [103] uses two separated networks, including a rigid transformation network and a deformable transformation network cascaded together for medical image registration.

Inspired by the above methods, we first propose a deep learning-based method for printed image registration based on VoxelMorph [101] in this subsection. The VoxelMorph method is a popular CNN network method for medical image registration, but we use it in the problem of printed image registration for the first time.

As shown in Figure 4.15, we first initialize the CNN weight parameters randomly by the Xavier initialization method [104]. The fixed image I_{fixed} and the moving image I_{moving} are used as the input to the CNN network. The output of the CNN is the deformation field ϕ. ϕ is a matrix with a size of $2 \times w \times h$, where w is the width and h is the height of the image. Similar to Subsection 4.2.2, the moving image I_{moving} is warped by the deformation field ϕ and is transformed to the moved image I_{moved}. The similarity loss L_{sim} is then used to compute the gradient $\frac{\partial L_{sim}}{\partial W}$ relative to the CNN weight parameters W. The weight parameters W are then updated by ΔW, which combines the learning rate λ and gradient $\frac{\partial L_{sim}}{\partial W}$.

Figure 4.16 shows the inference stage of deformable printed image registration. Consequently, the difference in the inference stage is that the parameters of CNN are fixed with the trained result and the moved image is output only once without the need for similarity loss computation. The main difference between the inference stage in Figure 4.16 and the training stage in Figure 4.15 is that there is gradient calculation and parameter backpropagation update in the training stage.

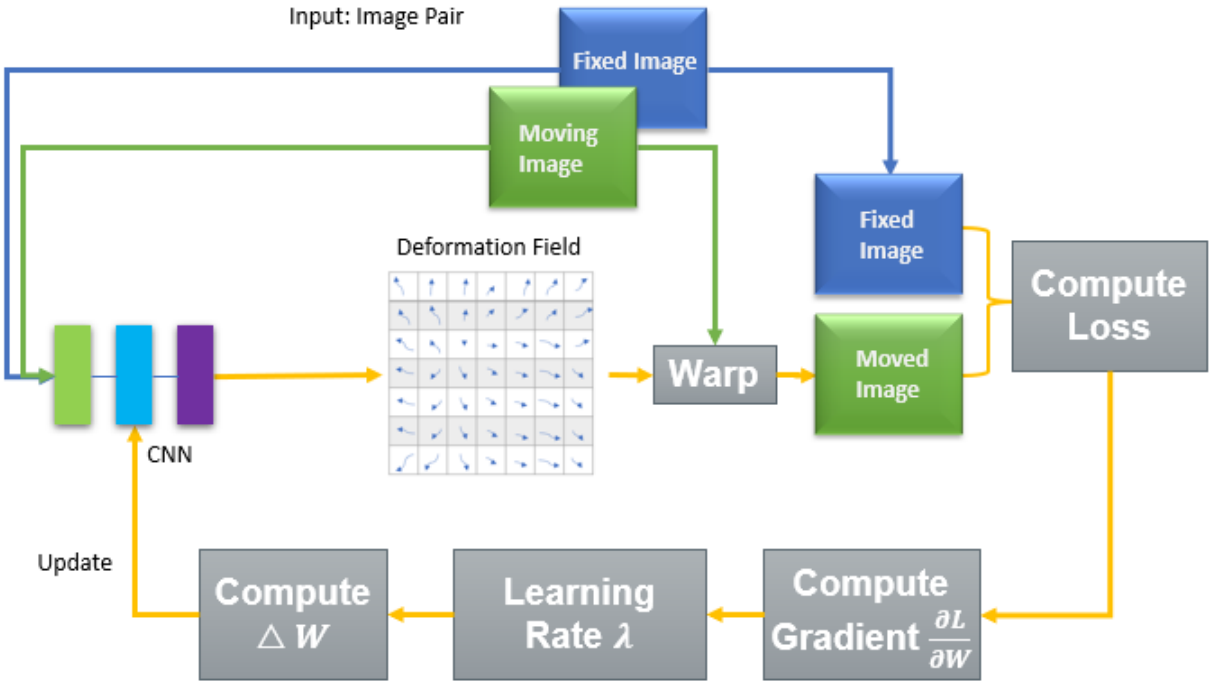


Figure 4.15. Deep unsupervised learning for deformable printed image registration.

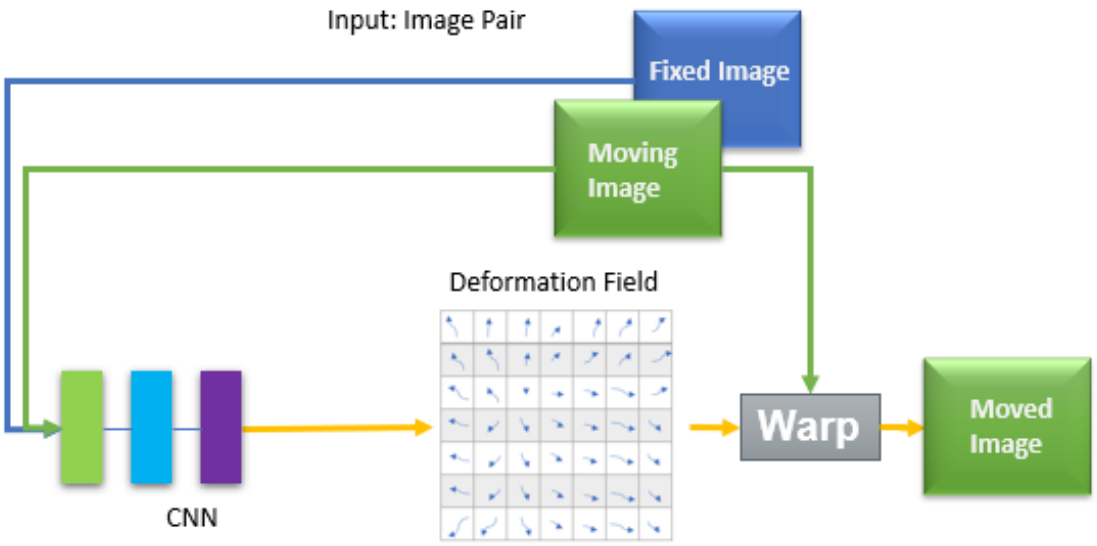


Figure 4.16. Inference stage for deep unsupervised learning for deformable printed image registration.

Figure 4.17 compares the global rigid image registration method and deep local deformable image registration method. The left figure is the global rigid registration process and the right figure is the local deformable image registration network process. As shown in the highlighted circle in Figure 4.17, the major difference is the generation of the deformation field ϕ. In the rigid registration, one set of parameters P and one transformation matrix T are used to generate the deformation field after updating iteratively. In the deformable registration, one deep CNN network (with weight W) is used to generate the deformation field.

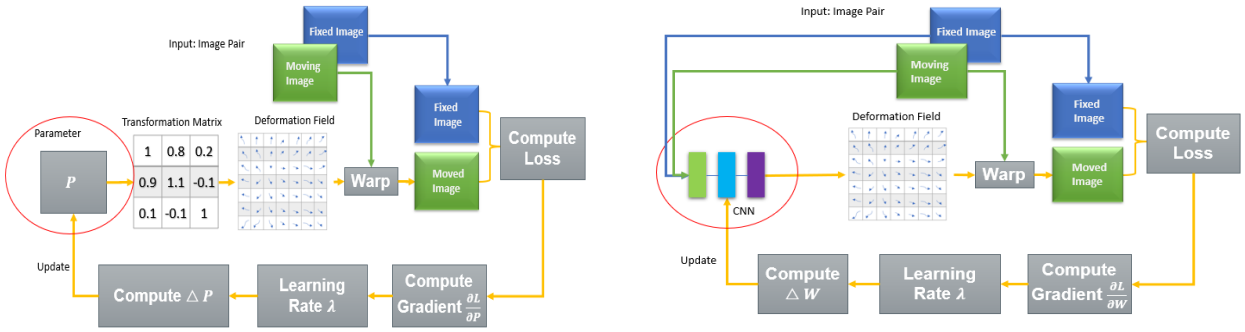


Figure 4.17. Comparison of global rigid registration and deep deformable registration.

In [101], the VoxelMorph method uses the U-Net [105] structure as its backbone network for 3-D medical image registration. For the 2-D deformable printed image registration problem in this dissertation, we continue to use the U-Net architecture. Figure 4.18 shows the U-Net architecture-based method for deformable printed image registration. The input to the U-Net is a tensor of a digital image (I_{fixed}) and a printed image (I_{moving}) concatenated together. The U-Net structure can maintain the same resolution for the output deformation field ϕ as the input images. Also, the hierarchical structure of the network is used for different levels of registration: higher-level features are used for overall larger movement of the images; lower-level features are used for the local smaller misalignment.

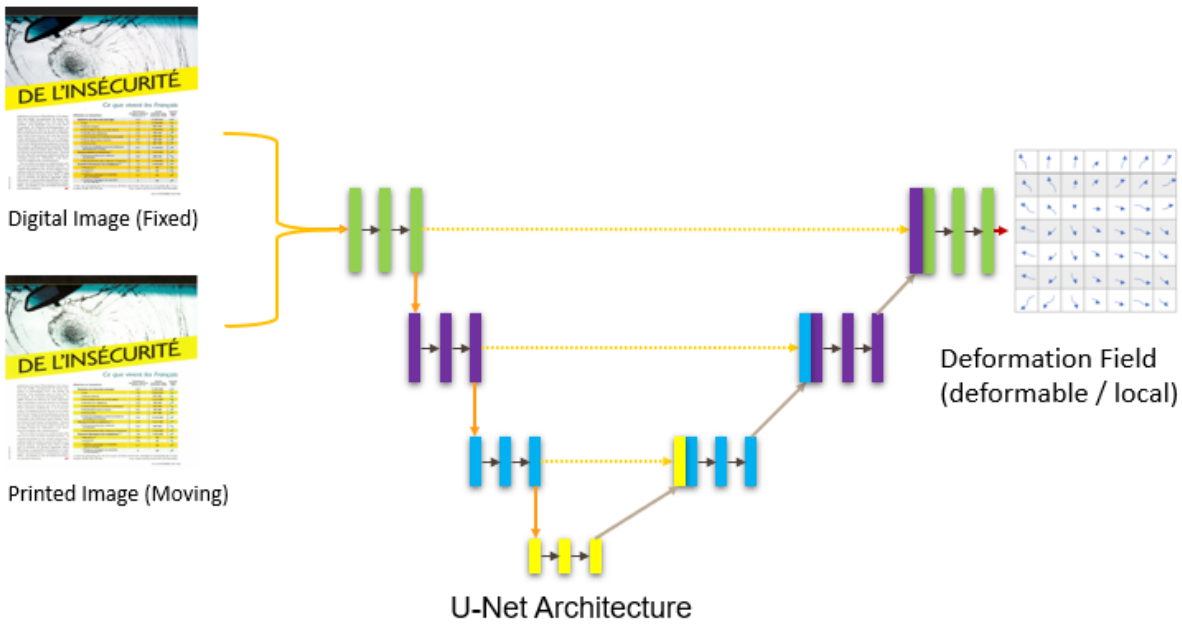


Figure 4.18. The U-Net architecture for deep unsupervised learning for deformable printed image registration.

Problem of U-Net VoxelMorph-based Method

The U-Net VoxelMorph-based method above can solve many local deformable printed image registration problems. For example, Figure 4.19 shows a qualitative result for printed image registration using the U-Net VoxelMorph-base architecture. The left figure shows that

the top part of the figure, including the straight line and characters, is misaligned, but the bottom part is well aligned. The right figure after registration has a better merged result.

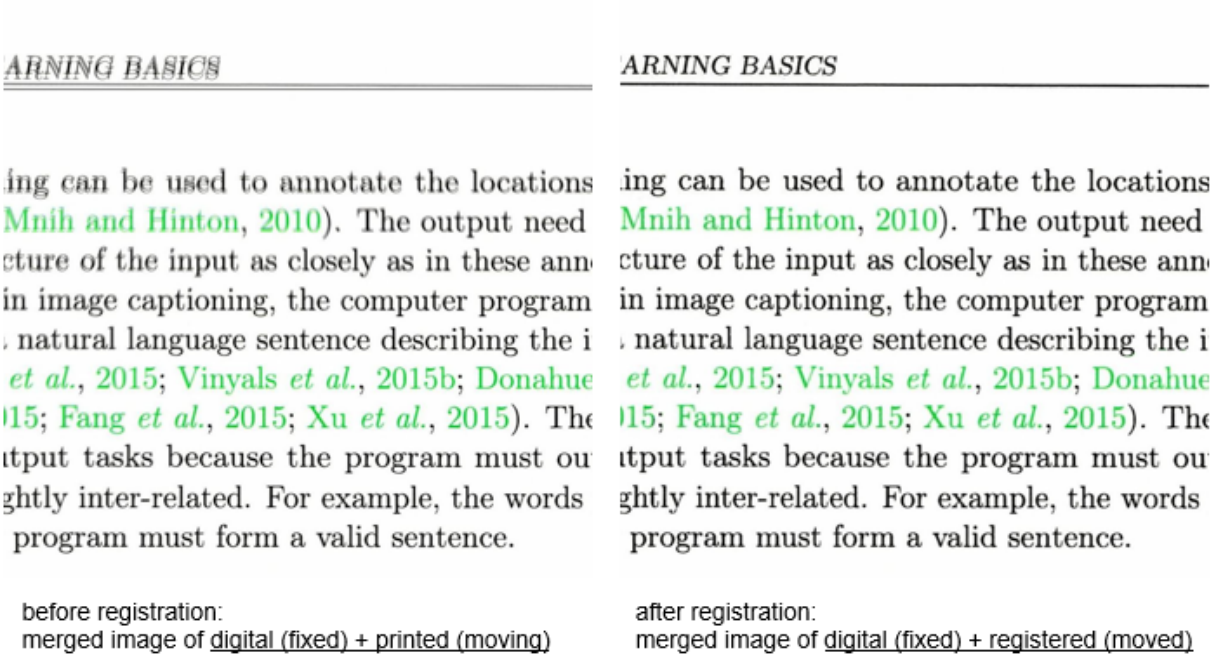


Figure 4.19. Qualitative result for deformable printed image registration using the U-Net VoxelMorph-base architecture.

However, there are still problems with the U-Net VoxelMorph-based method. For example, Figure 4.20 shows a failure case for deformable printed image registration using the U-Net VoxelMorph-based method. Some edge areas highlighted in the right figure have a diffused registration result.

The U-Net VoxelMorph-based method is also tested for medical image registration as shown in Figure 4.21. The test images for the experiment are retinal images from the Fundus Image Registration Dataset (FIRE) [106]. In Figure 4.21, the first image is the fixed image I_{fixed} before registration. There are key points for verifying the registration result in the first image. The second one is the moved image I_{moved} after registration. The third one is the comparison of prediction and ground truth. A detailed zoomed image of the third comparison is shown in 4.22. The fourth image is the deformation field ϕ. The vector size and direction in the deformation field represent the shift of each pixel. The colors of vectors represent different directions.

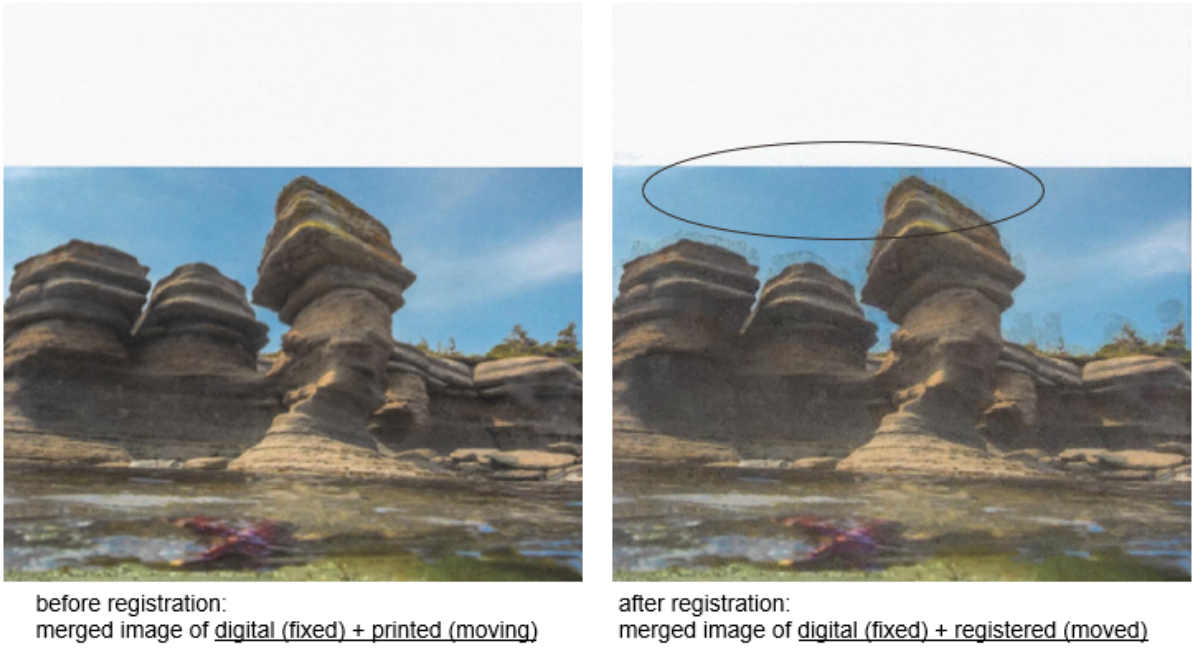


Figure 4.20. Failure case for deformable printed image registration using U-Net architecture.

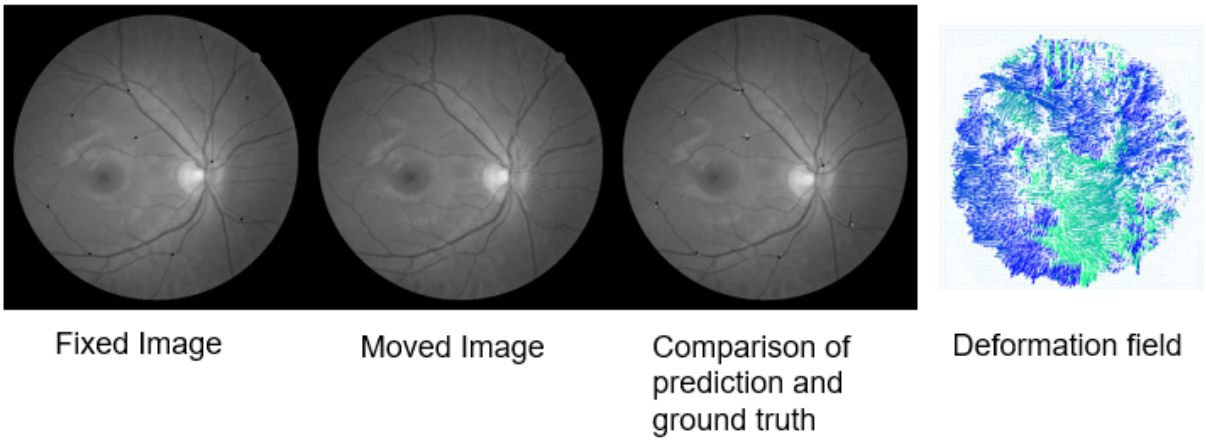


Figure 4.21. Failure case for deformable medical image registration using U-Net architecture.

As shown in Figure 4.22, the background image is the fixed image I_{fixed} before registration. The black dots are the key points for result verification. The straight lines represent the shift vectors of key points. In the figure, white lines are the predicted deformation vectors; black lines are the ground truth deformation vectors. For the same key point, we hope the direction and size of the predicted vector are close to the ground truth vector. From this example, we can see that the U-Net VoxelMorph-based method fails in this case.

Figure 4.22. Zoomed-in image of a failure case for deformable medical image registration. White lines are the predicted deformation vectors; black lines are the ground truth vectors.

In summary, there are several problems related to the U-Net VoxelMorph-based method:

- The deformation between two images is limited.

- The depth of U-Net requires additional architectural search or model fusion.
- The U-Net skip connection makes feature fusion only occur in the same resolution.
- The model takes up a lot of GPU memory for a large image input.
- The method is computationally intensive if using the U-Net VoxelMorph structure iteratively since weights of the network cannot be reused.

4.3.2 Recurrent Network-based Method (R-RegNet)

Because of the problems of the U-Net VoxelMorph-based method, we propose a Recurrent Registration Network, named R-RegNet, for printed image registration in this subsection by leveraging the correlation and recurrent architecture. The R-RegNet has a network structure similar to RAFT [107], but instead of estimating optical flow as in RAFT, the proposed R-RegNet is used for image registration. Figure 4.23 shows the overall process of the network. The fixed image I_{fixed} and moving image I_{moving} are concatenated together as the input to the feature extraction network. The weights of the feature extraction network for I_{fixed} and I_{moving} are shared to save computation time. The network structure of the feature extraction network is the same as the context extraction network. The feature of the fixed image F_{fixed} and the feature of the moving image F_{moving} are used in a correlation operation to generate 4 scale correlation volumes. The correlation volumes and the fixed image content features $F_{content}$ are then used iteratively as the input to the GRU blocks. The final output of the GRU blocks is the deformation field ϕ with the same resolution as the original images I_{fixed} and I_{moving}. The weight parameters W are then updated by backpropagation of the final unsupervised loss function L. Each block mentioned above will be introduced in detail in the following subsections.

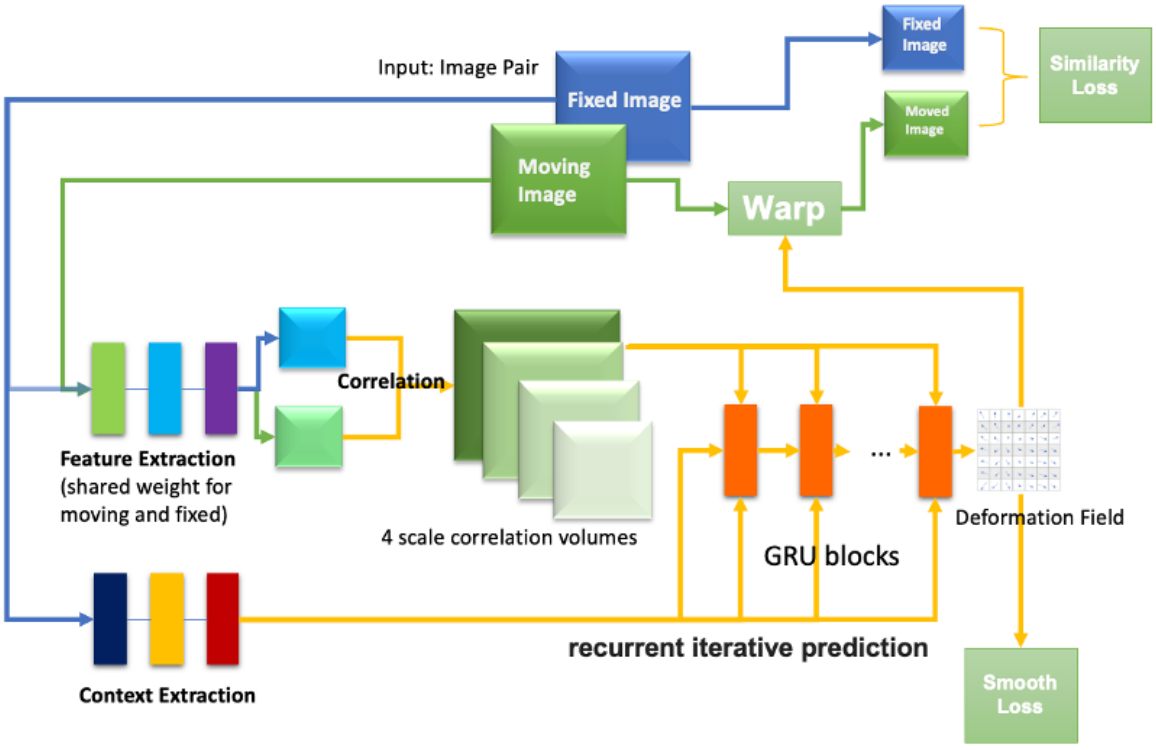


Figure 4.23. Proposed recurrent network structure R-RegNet for printed image registration.

Feature and Content Extraction

The feature extraction and content extraction modules have the same network structure. The extraction network is a modified residual network [62] with 6 residual blocks and 1 additional convolutional input layer and 1 additional output layer, as shown in Figure 4.24. I_{fixed} and I_{moving} are concatenated as the input of the feature extraction network to generate features with essential information such as edges, motions, contours, key points, etc. The network will automatically learn the feature in the training stage. Feature extraction can be considered as a dimension reduction process. The output features have a smaller resolution compared to original input images, saving computational memory and time. Because I_{fixed} and I_{moving} often have similar content, the network weights are shared to reduce backpropagation computation further. For the content extraction network, the input is the fixed

image I_{fixed} instead of I_{fixed} and I_{moving} together. The content feature $F_{content}$ is used as the reference feature when generating the deformation field ϕ.

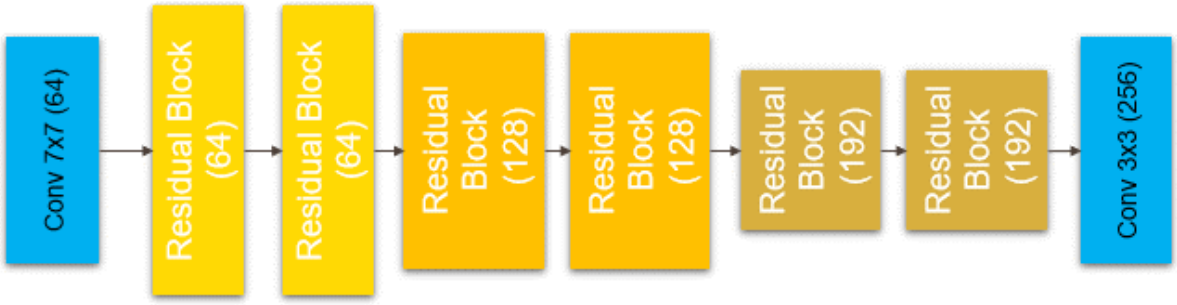


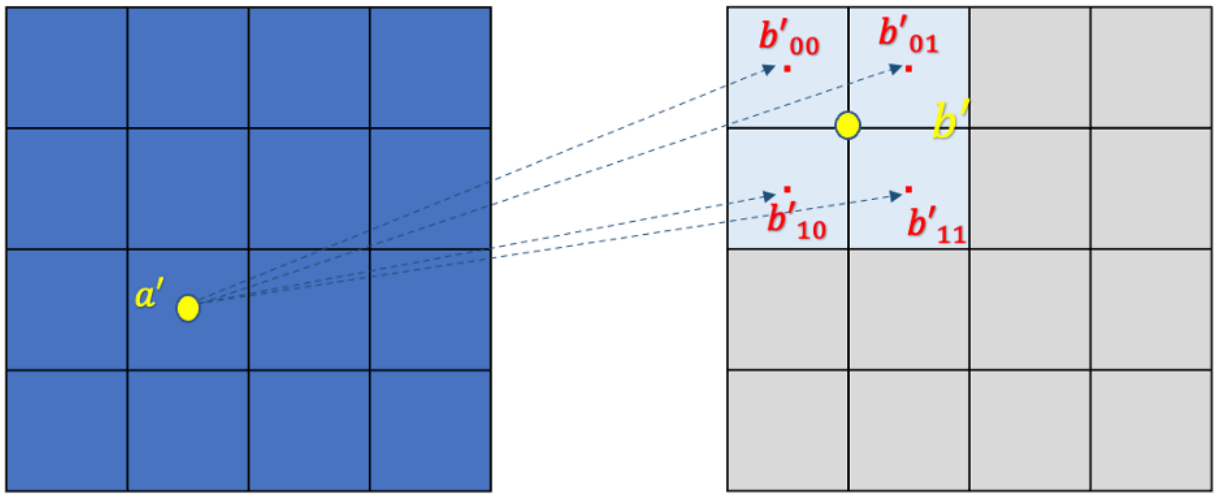
Figure 4.24. Feature and content extraction network.

Correlation Operation and Correlation Volumes

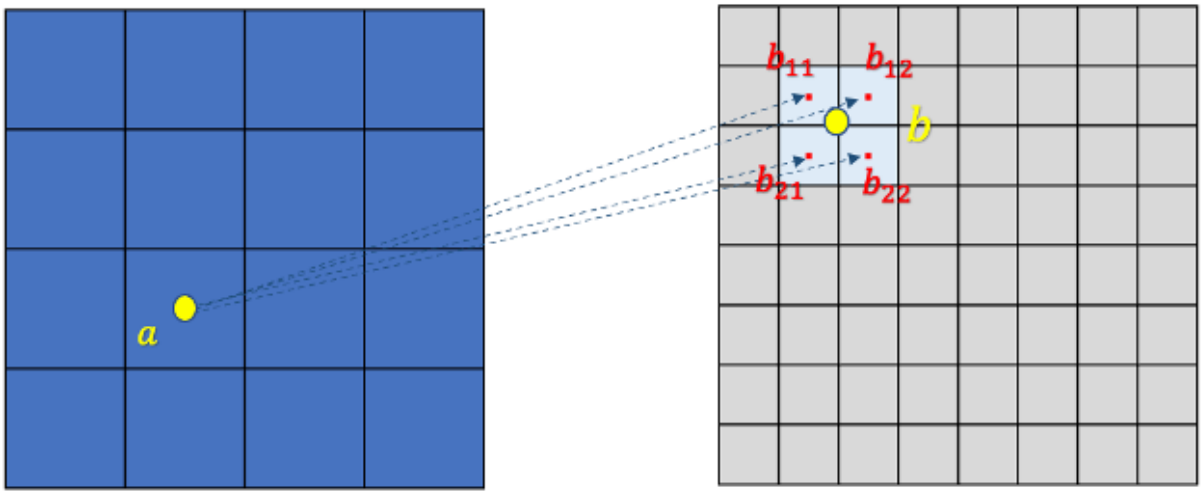
Correlation is a mathematical operation reflecting the similarity or dependency of two input features. It is also commonly used in deep neural networks for optical flow. Optical flow networks, such as Flownet [108], Flownet2 [109] and RAFT [107], usually take two or more consecutive frames as the inputs and predict the optical flow reflecting the movement between different video frames. The scheme between optical flow prediction and image registration is similar. The correlation and iterative prediction concepts can also be used in image registration.

In the U-Net VoxelMorph-based method, the U-Net skip connection makes features fusion only occur in the same resolution; and the receptive field of each pixel between two images is limited. In this work, we use a correlation operation for F_{fixed} and F_{moving} for different scales as shown in Figure 4.25. The left features are F_{fixed} and the right are F_{moving}. As shown in Figure 4.25a, F_{fixed} and F_{moving} have the same resolution. The correlation between features in position a' and b' is calculated as: $corr(a', b') = F(a')F(b'_{00}) + F(a')F(b'_{01}) + F(a')F(b'_{10}) + F(a')F(b'_{11})$. As shown in Figure 4.25b, the correlation between features in position a and b is calculated as: $corr(a, b) = F(a)F(b_{11}) + F(a)F(b_{12}) + F(a)F(b_{21}) + F(a)F(b_{22})$. By comparing Figure 4.25a and Figure 4.25b, we can find that the receptive field of $corr(a, b)$ around b is a

quarter of $corr(a', b')$. Different scales of correlation volumes can help generating deformation fields at different resolution levels. In the experiment, we choose 4 scale correlation volumes.



(a) F_{fixed} and F_{moving} have the same resolution.



(b) F_{moving} has 4 times resolution as F_{fixed}.

Figure 4.25. Correlation volumes for different scales. The left features are F_{fixed} and the right are F_{moving}.

Recurrent Neural Network (RNN) and Gated Recurrent Unit (GRU)

The Recurrent Neural Network (RNN) [110] is a type of deep neural network that reuses the same inner weight parameters and updates the new input from the previous layer output iteratively. RNN is commonly used for natural language processing (NLP) [111], video

processing [112], speech recognition [113] and time series prediction [114] tasks. The image registration task is an iterative process that has similarities with RNN structure tasks. Also, the image features can be used as the hidden states in the RNN during iterations without re-calculation for feature extraction, content extraction and correlation volumes. Compared to the U-Net VoxelMorph-based method, the length of iteration for RNN is flexible. So the deformation field ϕ between two input images is not limited.

In this work, we use a specific type of RNN named Gated Recurrent Unit (GRU) [115]. It has the property similar to that of long short-term memory (LSTM) [116]. Compared with LSTM, the GRU has no output gate since the output gate is mainly designed for none or empty signal output in time series tasks. So the GRU will always generate a valid output for the deformation field ϕ. Also, the GRU has fewer parameters than LSTM.

As shown in Figure 4.26, the correlation volumes feature and the content feature is concatenated to a vector as the input $x(t)$. $z(t)$ is the update gate vector and $h^*(t)$ is the candidate activation vector. The hidden state $h(t)$ refers to the deformation feature in step t. The hidden state $h(t-1)$ of the previous step $t-1$ will be used as the input $h(t)$ of step t.

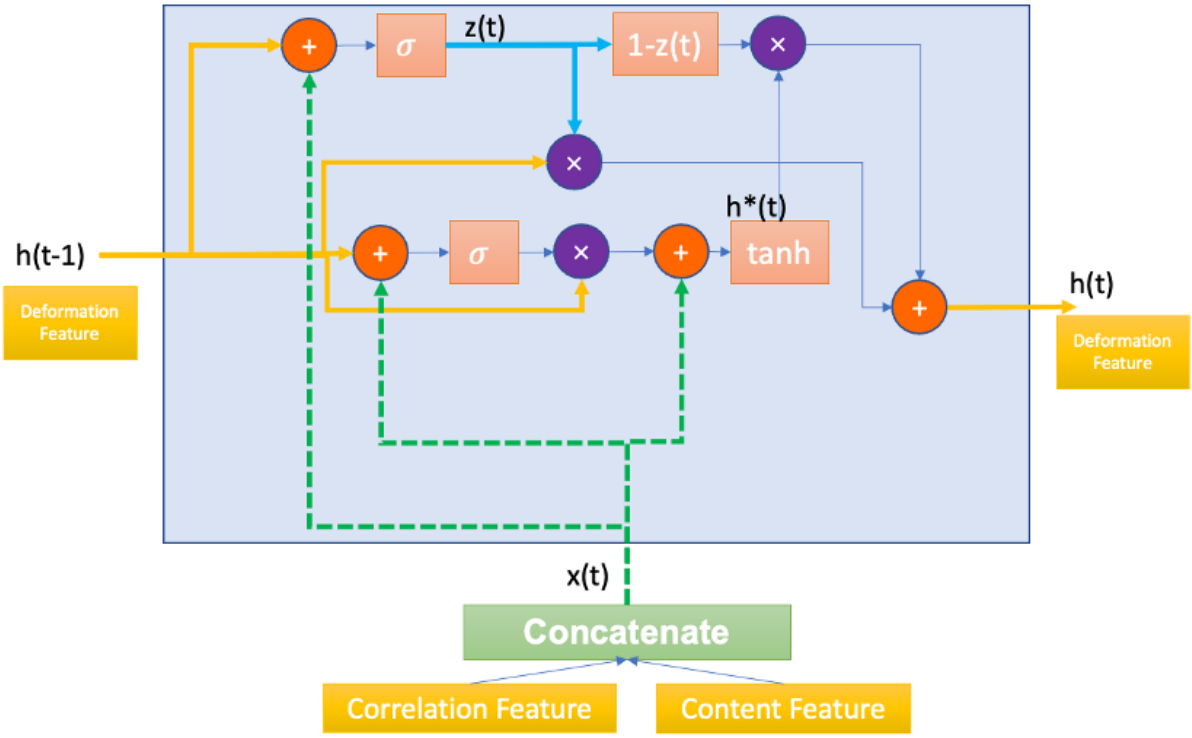


Figure 4.26. Gated Recurrent Unit (GRU).

4.3.3 Unsupervised Loss Function

The final unsupervised loss function consists of three parts: similarity loss L_{sim}, smooth loss L_{smooth} and reconstruction loss L_{recon}:

$$L = L_{sim}(I_{fixed}, I_{moved}) + L_{smooth}(\phi) + L_{recon}(\phi_1, \phi_2) \tag{4.1}$$

Similarity Loss

Similarity loss is a function to measure the similarity between the moved image I_{moved} and the fixed image I_{fixed}. I_{moved} is the warped moving image: $I_{moved} = \phi \circ I_{moving}$. So the similarity loss can be expressed as:

$$L_{sim}(I_{fixed}, I_{moved}) = L_{sim}(I_{fixed}, \phi \circ I_{moving}) \tag{4.2}$$

For example, Mean Square Error (MSE) or Normalized Cross Correlation (NCC) can be used for the similarity loss function.

Mean Square Error (MSE) is a function to measure the square error between two features.

$$MSE(X,Y) = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}(X_{\mathrm{i}} - Y_{\mathrm{i}})^2 \tag{4.3}$$

$$L_{MSE}(I_{fixed}, I_{moved}) = MSE(I_{fixed}, I_{moved}) \tag{4.4}$$

Normalized Cross Correlation is a normalized loss function to measure the relative displacement of one feature to another feature vector. The NCC is defined as:

$$NCC(X,Y) = \frac{1}{N-1}\sum_{\mathrm{i}=1}^{N}\frac{(X_{\mathrm{i}} - \mu_X)(Y_{\mathrm{i}} - \mu_Y)}{\sigma_X \sigma_Y} \tag{4.5}$$

Because higher NCC indicates a better alignment result, the final loss function is the negative of NCC.

$$L_{NCC}(I_{fixed}, I_{moved}) = -NCC(I_{fixed}, I_{moved})$$

For printed image registration problems, the printing and scanning processes cause a difference in color space between the fixed and moved image. To reduce the impact of the color difference problem, choosing NCC loss instead of MSE loss is better.

Smooth Loss

Overfitting occurs if the networks learn only to minimize the similarity loss and introduce a non-smooth deformation field during training. As shown in Figure 4.27, we hope the deformation field ϕ has similar directions and magnitudes in the neighborhood area.

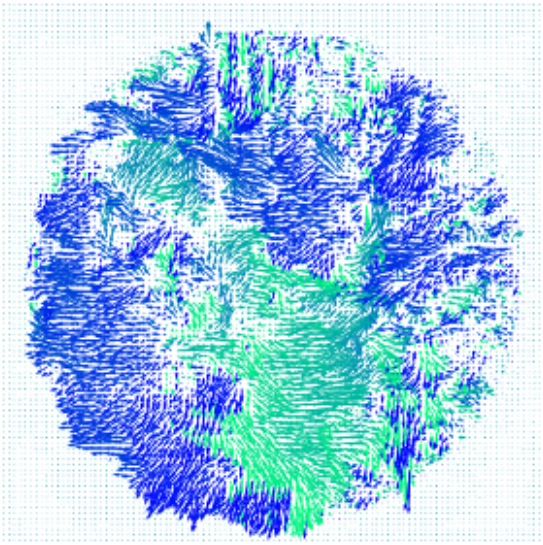


Figure 4.27. Example of a deformation field ϕ.

The smooth loss is introduced to force the consistency of the deformation field by minimizing the total absolute difference of the nearby deformation vectors, as shown in Equation 4.6.

$$L_{smooth}(\phi) = \sum_{\mathrm{i}=1}^{N} \nabla v_{\mathrm{i}} \tag{4.6}$$

Here, N is the total number of pixels and ∇v_{i} is the difference between the neighborhood deformation vectors. The neighborhood size depends on the average deformation magnitude in different use cases. Usually, the neighborhood is a square of 3×3 or 9×9 for printed image registration problems.

Reconstruction Loss

For deformable image registration, the deformation field ϕ should be symmetric when the order of I_{fixed} and I_{moving} changed. For example, we define the deformable registration network as a function $F(\cdot, \cdot)$. ϕ_1 is the deformation field output from input I_{fixed} and I_{moving}, as shown in Equation 4.7.

$$\phi_1 = F(I_{fixed}, I_{moving}) \tag{4.7}$$

Then we change the order of input to the registration networks function F. ϕ_2 is the output from input I_{moving} and I_{fixed}, as shown in Equation 4.8.

$$\phi_2 = F(I_{moving}, I_{fixed}) \tag{4.8}$$

As shown in Figure 4.28, the top pipeline has the fixed image as the first input and the moving image as the second input. The bottom pipeline changes the input order to moving image and fixed image. The deformation field ϕ_1 is the output of the top pipeline; the deformation field ϕ_2 is the output of the bottom pipeline. For the image registration problem, we can add the one-to-one mapping constraint so that $\phi_1 \approx \phi_2^{-1}$, which means the offset of the deformation field should be opposite.

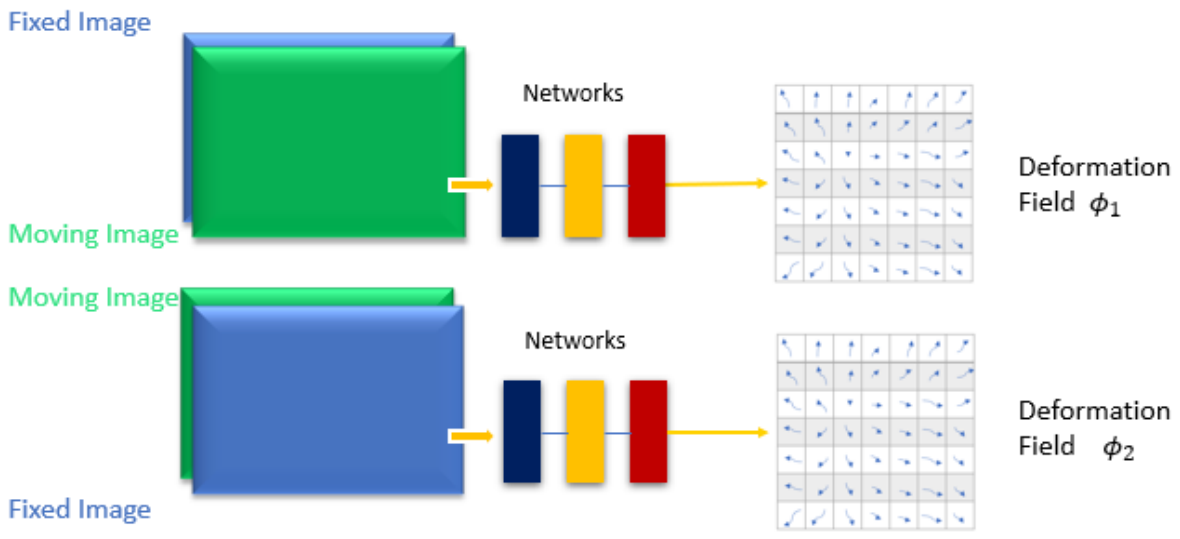


Figure 4.28. Unsupervised loss function: reconstruction loss.

The reconstruction loss is proposed in this work to force $\phi_1 \longrightarrow \phi_2^{-1}$ in the network training process, as shown in Equation 4.9. The reconstruction loss is used in the training of the network for the consistency of the registration output.

$$L_{recon}(\phi_1, \phi_2) = sim(\phi_1, \phi_2^{-1}) = (\phi_1 - \phi_2^{-1})^2 \tag{4.9}$$

4.3.4 Data Preparation

SIMULATED Dataset

For the experiment, we simulate a dataset for printed image pairs. The goal for the simulation is to make changes, such as slight movement, color shift, noise change, etc., for the digital image and generate images similar to the printed version. 4030 images are collected from the Fliker dataset. For each original image, 10 image pairs are simulated. There are 40300 image pairs in total (32240 for the training set and 8060 for the test set). The simulation is not based on the physics world for printers and scanners. Instead, data augmentation methods used for the simulation include cropping, saturation removal, piece-wise affine transformation, elastic transformation, noise and color shift. Example images in the SIMULATED dataset are shown in Figure 4.29.

HPLAB Dataset

In this work, we also collected a real printed and scanned dataset, named HPLAB DATASET, as shown in Figure 4.30. The original resolution ranges from 994×994 to 2330×3179. There are 332 image pairs in total (239 for the training set and 93 for the test set). Each image pair consists of a fixed (digital) image I_{fixed} and a moving (printed) image I_{moving}.

The images are cropped with a stride of 300 pixels. In the training set, there are 20994 images cropped from original data. In the test set, there are 8120 cropped images as shown in Figure 4.31.

Figure 4.29. SIMULATED DATASET: digital and simulated pairs.

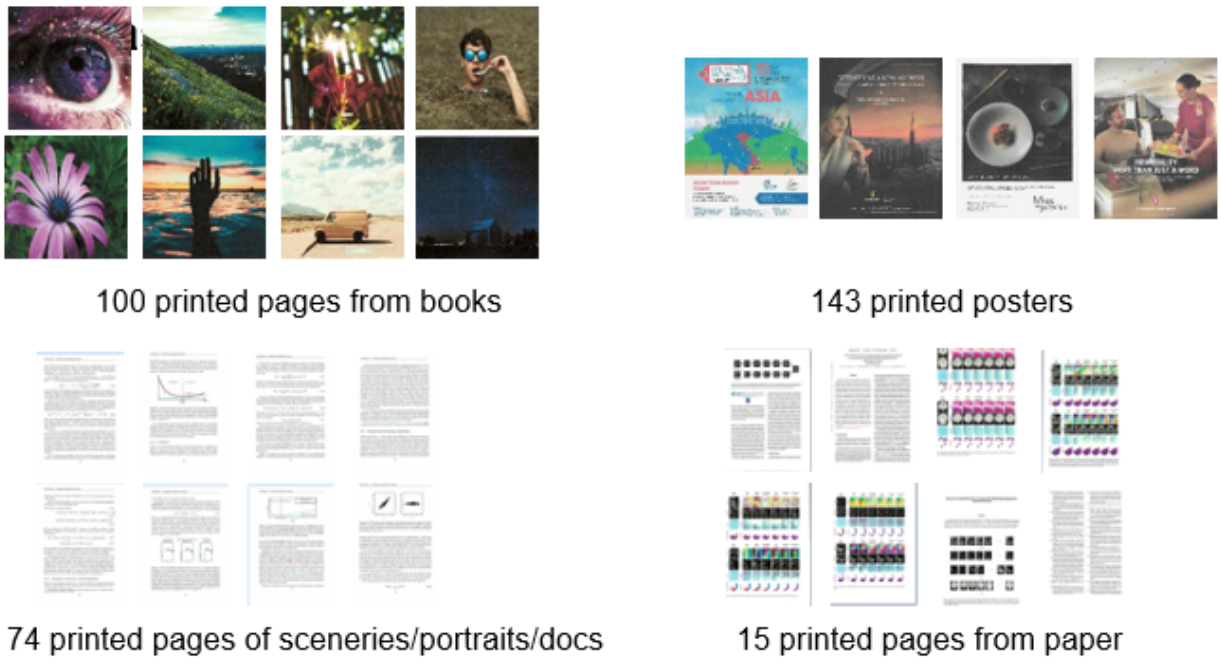


Figure 4.30. HPLAB DATASET: digital and printed pairs.

Figure 4.31. HPLAB DATASET: example of cropped image pairs.

FIRE Dataset

The image registration performance on the HPLab dataset can only be measured by MSE or other similarity metrics since no landmark ground truth is provided in the dataset. Another dataset named Fundus Image Registration Dataset (FIRE) [106] is used for an additional experiment with ground truth, as shown in Figure 4.32. The white dots in the figure are the key control points for result verification. The FIRE dataset consists of 134 retinal image pairs. The ground truth has control point movement vectors from the fixed image to the moving image. We resized the original resolution 2912×2912 to 512×512. The error is calculated as the mean of the distance between registered and fixed images.

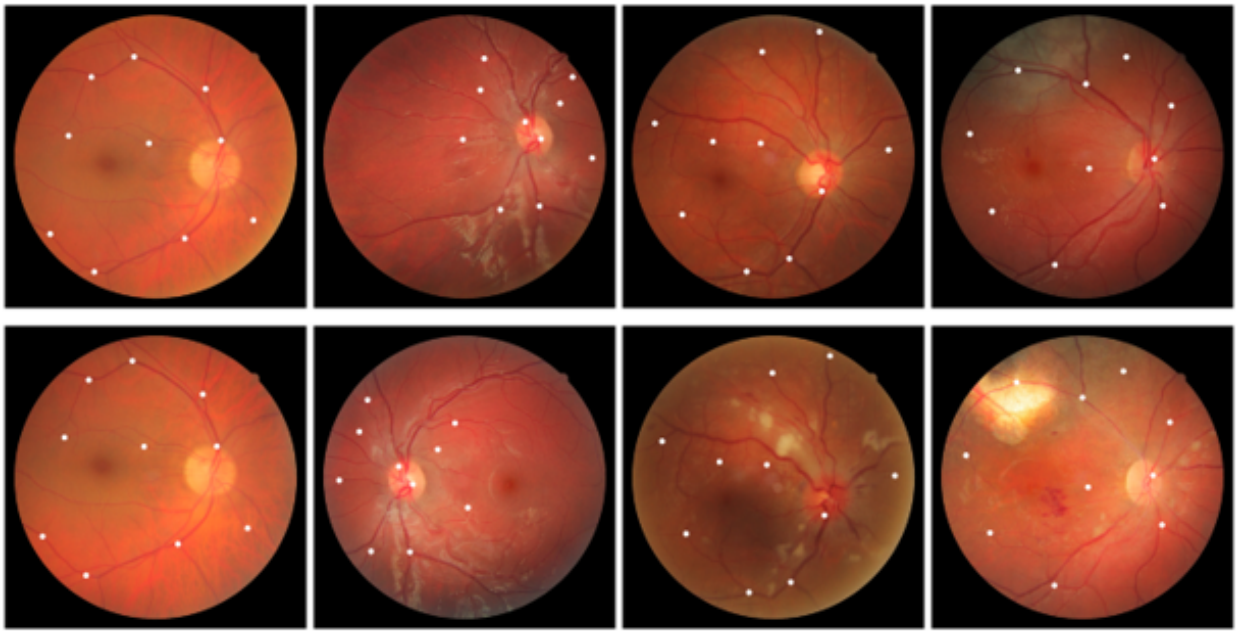


Figure 4.32. Retinal images example from FIRE DATASET.

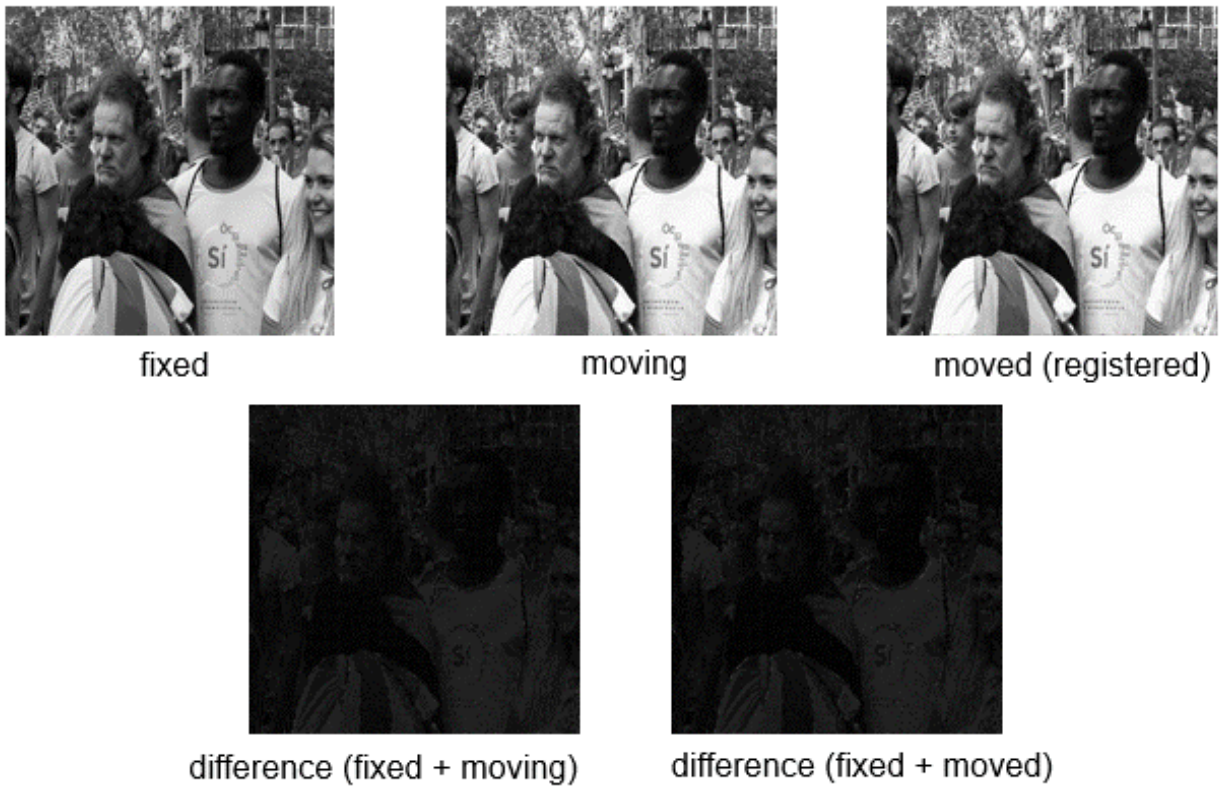
4.3.5 Experimental Results

This section introduces experimental results between the U-Net VoxelMorph-based method and the proposed recurrent network R-RegNet method for the different datasets mentioned above.

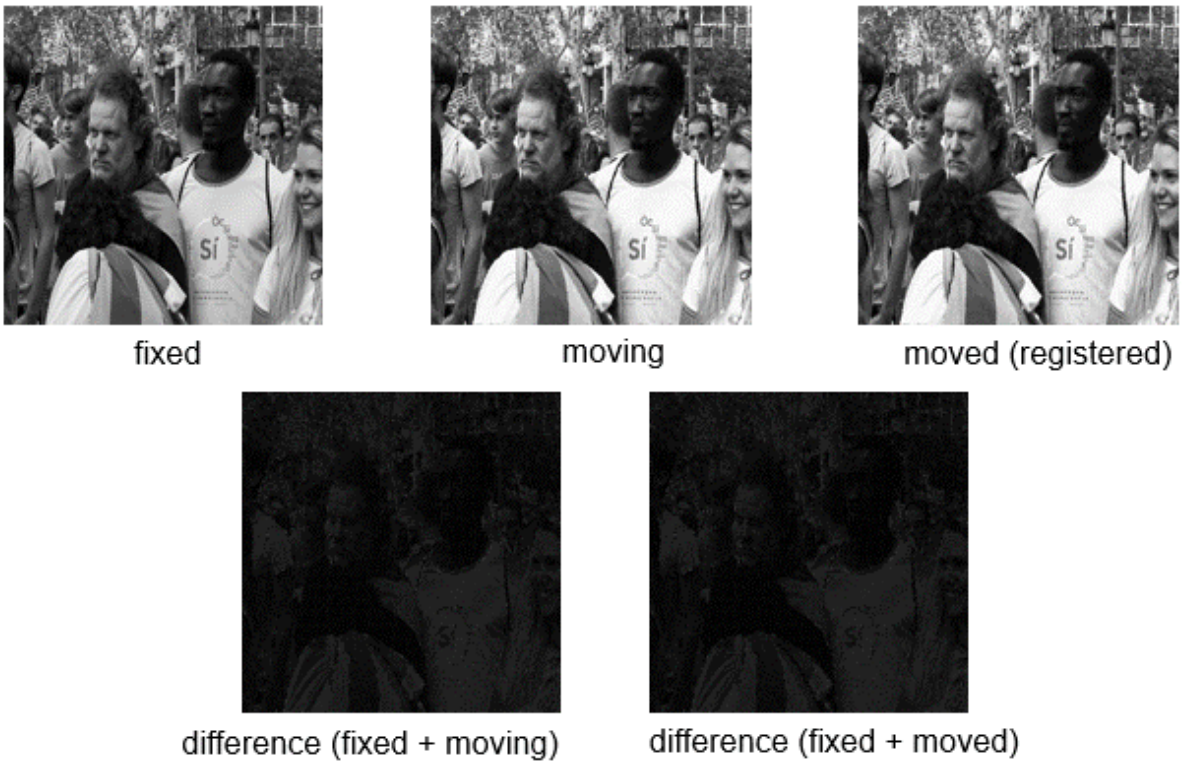
SIMULATED Dataset

Figure 4.33a shows the test result of U-Net VoxelMorph-based method. Figure 4.33b shows the test results of proposed recurrent network R-RegNet method.

Figure 4.34 is the detailed difference comparison of the test results of the U-Net VoxelMorph-based method and the proposed R-RegNet method. As shown in the comparison image, the difference image of the proposed R-RegNet method is slightly darker compared to the U-Net VoxelMorph-based method. A lower absolute difference value (or a darker difference image) means a better registration result.



(a) Test result of U-Net VoxelMorph-based method.



(b) Test result of the proposed R-RegNet method.

Figure 4.34. Comparison of the test results of the U-Net VoxelMorph-based method and the proposed R-RegNet method.

Table 4.1 also shows a lower average RMSE score of all test images for the proposed recurrent network R-RegNet than the U-Net VoxelMorph-based method. The RMSE is calculated as $RMSE = \sqrt{\frac{1}{n}\Sigma_{\mathrm{i}=1}^{n}\left(I_{\mathrm{i}} - \tilde{I}_{\mathrm{i}}\right)^2}$, similar to Chapter 2, where $\tilde{I}_{\mathrm{i}}$ is the pixel value of coordinate i after registration, I_{i} is the pixel value of coordinate i in the Fixed image and n is the total number of pixels. A lower RMSE score means the value of the same pixel coordinate is closer between the fixed and the moved image, which also means a better registration result.

Table 4.1. Comparison of similarity scores of the U-Net VoxelMorph-based method and the proposed R-RegNet method for SIMULATED dataset

	U-Net VoxelMorph-Based Method	R-RegNet Method
RMSE score	18.975	18.916

Figure 4.35 is a failure case of the proposed R-RegNet method in the SIMULATED test Dataset. The left one is the fixed image. The middle one is the moving image before registration. There is some distortion in the moving image compared to the fixed image. For

example, the capitalized character “S” on the cloth is distorted in the middle image. The registration algorithm is expected to correct this kind of distortion. However, on the right side, the moved image after registration still shows the distortion, which means the proposed R-RegNet method fails to register two input images.

Figure 4.35. A failure case in the test result of the proposed R-RegNet method.

Figure 4.36 is a detailed comparison of the difference image before and after registration. As shown in the comparison image, the difference image after registration is still similar to the image before registration.

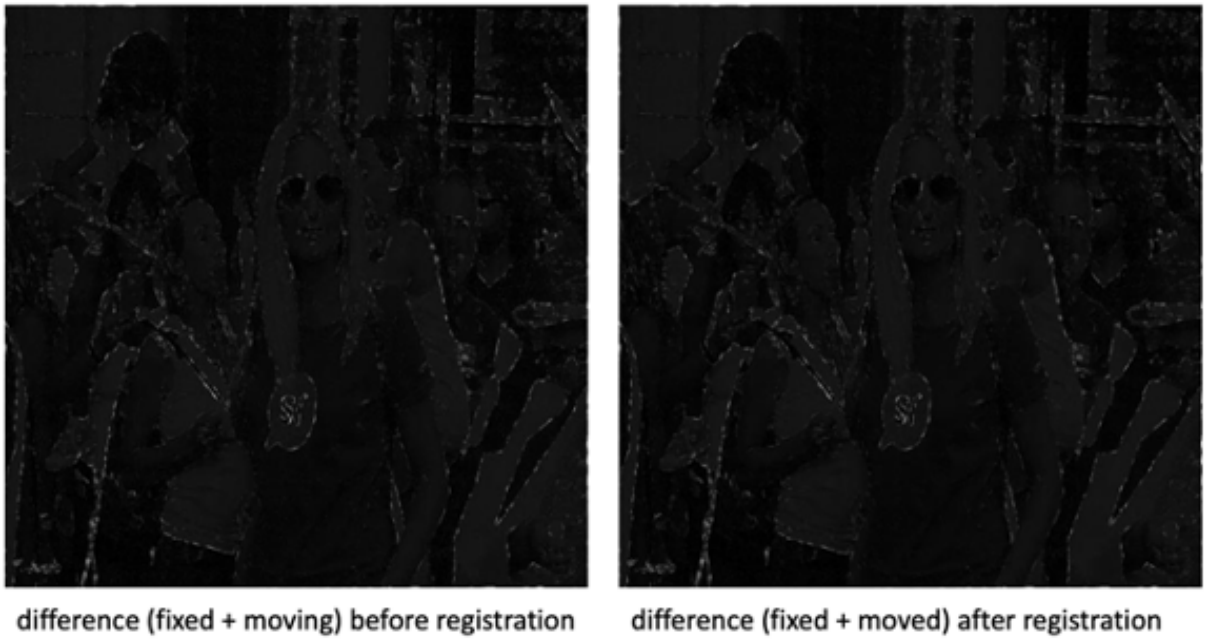


Figure 4.36. A failure case: comparison of the the difference images before and after the proposed R-RegNet registration.

HPLAB Dataset

Figure 4.37 and Figure 4.38 are two examples that compare the test result of the U-Net VoxelMorph-based method and the proposed R-RegNet method. Similar to the previous subsection, a lower absolute difference value (or a darker difference image) means a better registration result. The comparison images show that the absolute difference images of the proposed R-RegNet method are better than the U-Net VoxelMorph-based method.

Figure 4.37. Comparison 1 of test result of the U-Net VoxelMorph-based method and the proposed R-RegNet method.

Figure 4.38. Comparison 2 of test result of the U-Net VoxelMorph-based method and the proposed R-RegNet method.

Table 4.2 is a comparison of RMSE score in HPLAB test dataset. The table also shows that the proposed R-RegNet method is better than the U-Net VoxelMorph-based method by RMSE score.

Table 4.2. Comparison of similarity scores of the U-Net VoxelMorph-based method and the proposed R-RegNet method for HPLAB dataset

	U-Net VoxelMorph-Based Method	R-RegNet Method
RMSE score	24.123	23.899

In Figure 4.39, we also show a failure case of the proposed R-RegNet method in the HPLAB test Dataset. The left one is the fixed image. The middle one is the moving image before registration. And the right one is the moved image after registration. The post of the door is curved unnaturally after registration in the moved image.

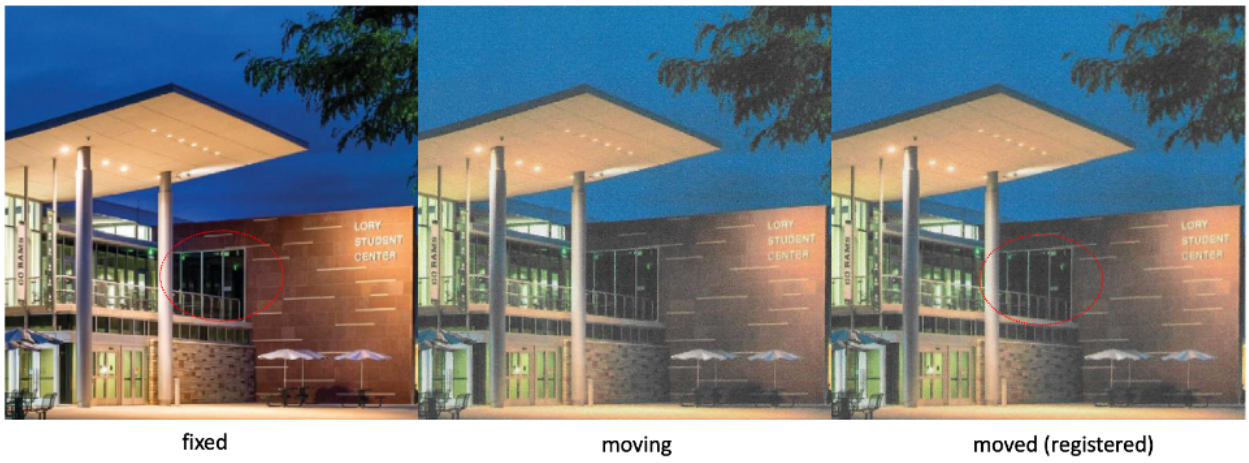


Figure 4.39. A failure case of the proposed R-RegNet method in the HPLAB test dataset result.

FIRE Dataset

For FIRE dataset mentioned in Section 4.3.4, we have 4 experiments for comparison. The first experiment trains the U-Net VoxelMorph-base network on the FIRE dataset. The second experiment trains the proposed R-RegNet on the FIRE dataset. The third experiment uses a pre-trained FlowNet [108]. We choose the FlowNetS model used in the original paper,

which has been pre-trained on the Flying Chairs dataset with 22,872 image frames. The fourth experiment re-trains the pre-trained FlowNet on the FIRE dataset.

Figure 4.40 and 4.41 are qualitative result of U-Net VoxelMorph-based method, the proposed R-RegNet method, the pre-trained FlowNet method, and the re-trained FlowNet method in the FIRE dataset.

In Figure 4.40, the first row is the U-Net method result and the second row is the proposed R-RegNet method result. The third row is the pre-trained FlowNet result and the last row is the re-trained FlowNet result, respectively. The first column is the fixed image I_{fixed} before registration. The black dots are key control points for verifying the registration result. The second column is the moved image I_{moved} after registration. The third column, which is shown in Figure 4.41 in detail, is the comparison of prediction and ground truth. The fourth column is the deformation field ϕ. The vector size and direction in the deformation field represent the shift of each pixel. The colors of vectors represent different directions.

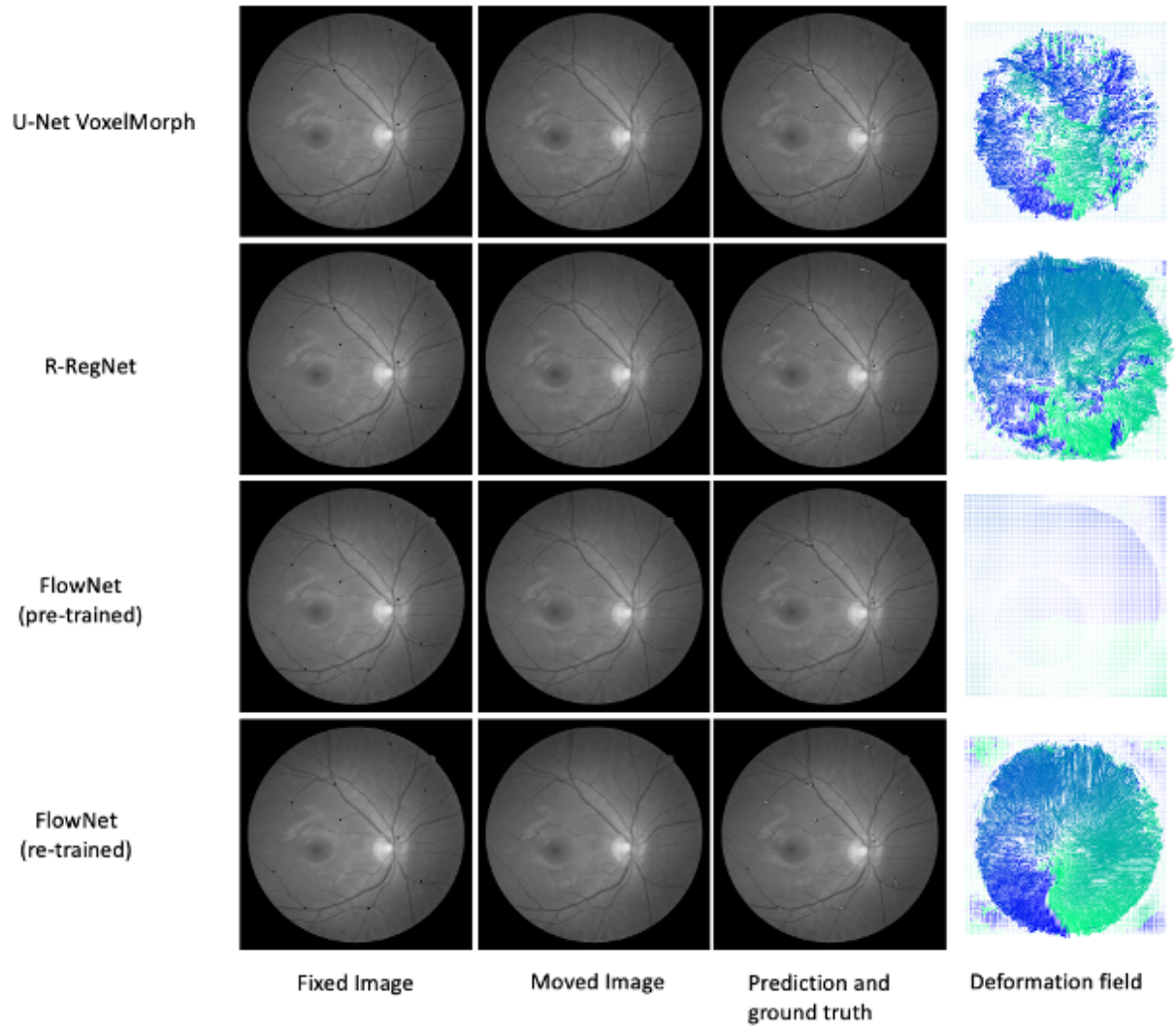


Figure 4.40. Comparison of test result of the U-Net VoxelMorph-based method, the proposed R-RegNet method, the pre-trained FlowNet method, and the re-trained FlowNet method.

As shown in Figure 4.41, the background image is the fixed image I_{fixed} before registration. The black dots are the key control points for result verification. The straight line represents the shift vector of each key point. In Figure 4.41, white lines are the predicted deformation vectors and black lines are the ground truth deformation vectors, respectively. We hope the direction and size of the predicted vectors are close to the ground truth vectors for the same key point. The result shows that the proposed R-RegNet method outperforms

the other 3 methods. The result also shows that the FlowNet is not working even when pre-trained with 22,872 images on the other dataset.

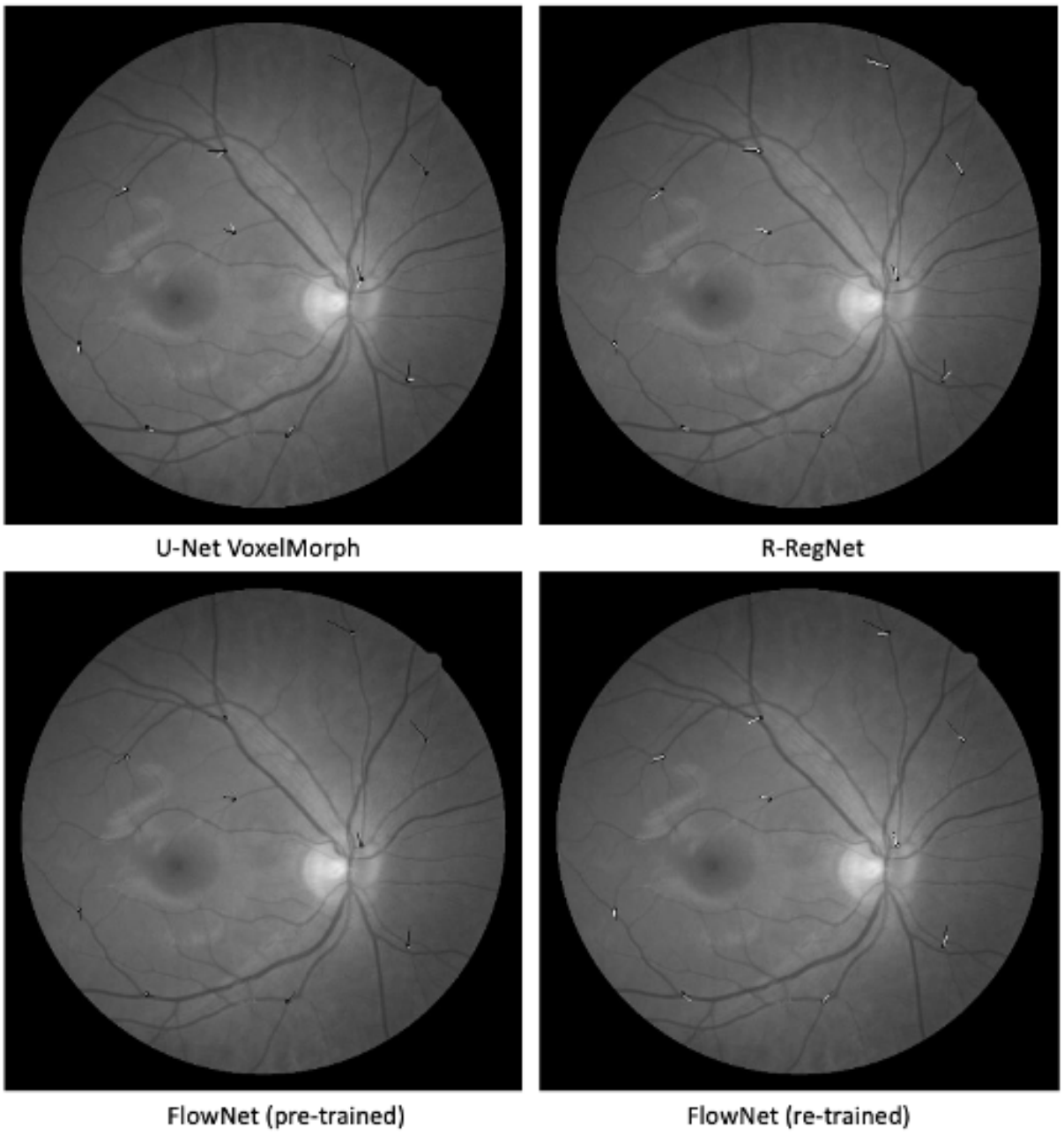


Figure 4.41. Detailed comparison of the control points movement. White lines are the predicted deformation vectors; black lines are the ground truth vectors.

In Table 4.3, the average error between ground truth and predicted deformation on control points proves the advantage of the proposed R-RegNet method. The average error is calculated as:

Average Error $= \frac{1}{n}\Sigma_{\mathrm{i}=1}^{n}\frac{\Sigma_{\mathrm{j}=1}^{k}||\phi_{\mathrm{j}}-\tilde{\phi}_{\mathrm{j}}||}{k}$

where $\tilde{\phi}_{\mathrm{j}}$ is the predicted deformation vector from the registration network, ϕ_{j} is the ground truth deformation vector, k is the number of key control points, and n is the total number of tests in the dataset. A lower Average Error score means a better registration result.

Table 4.3. Comparison of the similarity scores of the U-Net VoxelMorph-based method, the proposed R-RegNet method, the pre-trained FlowNet method, and the re-trained FlowNet method

Method	**Average Error (pixels)**
U-Net VoxelMorph	10.050
R-RegNet	7.737
FlowNet (pre-trained)	11.660
FlowNet (re-trained)	8.225

Figure 4.42 shows a failure corner case of the proposed R-RegNet method. As shown in the figure, the first column is the fixed image I_{fixed} before registration. The second column is the moved image I_{moved} after registration. The third column is the comparison of prediction and ground truth. In the third column, white lines are the predicted deformation vectors; black lines are the ground truth deformation vectors. The fourth column is the deformation field ϕ. From Figure 4.42, we can find that only 4 out of 10 predicted vectors are close to the ground truth vectors (difference of direction less than 30 degrees and difference of length less than 20%).

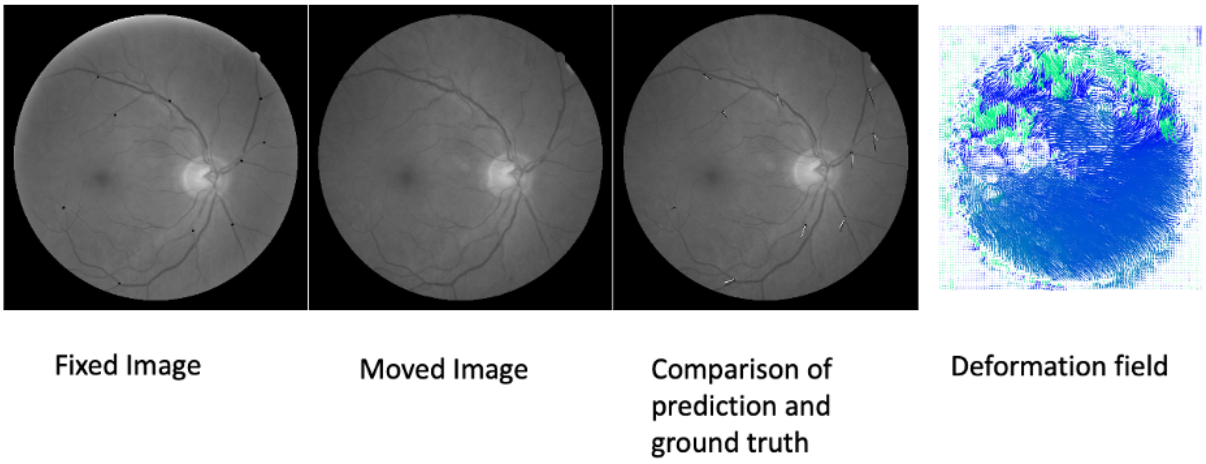


Figure 4.42. A failure case of the proposed R-RegNet method in FIRE test dataset.

4.4 Summary of Contributions

In this section, we propose to use the deep learning framework PyTorch for intensity-based image registration for parallel computation acceleration. We implemented the U-Net VoxelMorph-based method for deformable printed image registration. Then, we further propose the R-RegNet method for deformable printed image registration. The proposed recurrent network-based method R-RegNet uses an integrated framework to solve image registration for multi scales simultaneously. The proposed R-RegNet uses one weight-sharing network for feature and content extraction, which reduces the model size. The weight sharing of GRU could also speed up the iterative training process. We also introduce an unsupervised loss function including similarity loss L_{sim}, smooth loss L_{smooth} and reconstruction loss L_{recon}. The proposed reconstruction loss is used to constrain the consistency of the registration output. The reconstruction loss can train the system to generate a more robust network faster. We create the SIMULATED dataset for experiments, using data augmentation to simulate printed images. We also create the HPLAB dataset with real digital and printed pairs for the image registration problem. By comparison, the experimental results show that the R-RegNet method outperforms other methods in terms of mean square error and average ground truth deformation error.

5. PHOTOREALISTIC IMAGES SIMULATION FOR 6D POSE ESTIMATION

5.1 Introduction

This chapter introduces a new image dataset for object detection and 6D pose estimation, named Extra FAT. The dataset consists of 825K photorealistic RGB images with annotations of ground-truth location and rotation for both the virtual camera and the objects. A registered pixel-level object segmentation mask is also provided for object detection and segmentation tasks. The dataset includes 110 different 3D object models. The object models were rendered in five scenes with diverse illumination, reflection, and occlusion conditions.

Pose estimation of surrounding objects serves as the basis of various computer vision applications such as virtual reality (VR), augmented reality (AR), robotic manipulation, autonomous navigation, and human-machine interaction. For example, a virtual object should be accurately registered with the real world to insert it in an AR application. To do so, the geometry, pose and shape of the objects and the surfaces composing a scene need to be inferred from the image, video or depth information. For tasks such as autonomous navigation or robot manipulation, the pose of the objects needs to be estimated to move the robot or vehicle properly. In order to understand the geometry and position of the objects composing a scene, object detection and pose estimation techniques are required.

Figure 5.1. Each frame in the Extra FAT dataset consists of an image with 640×480 resolution, a registered pixel-level object segmentation mask, and the pose ground truth of the virtual camera and the objects

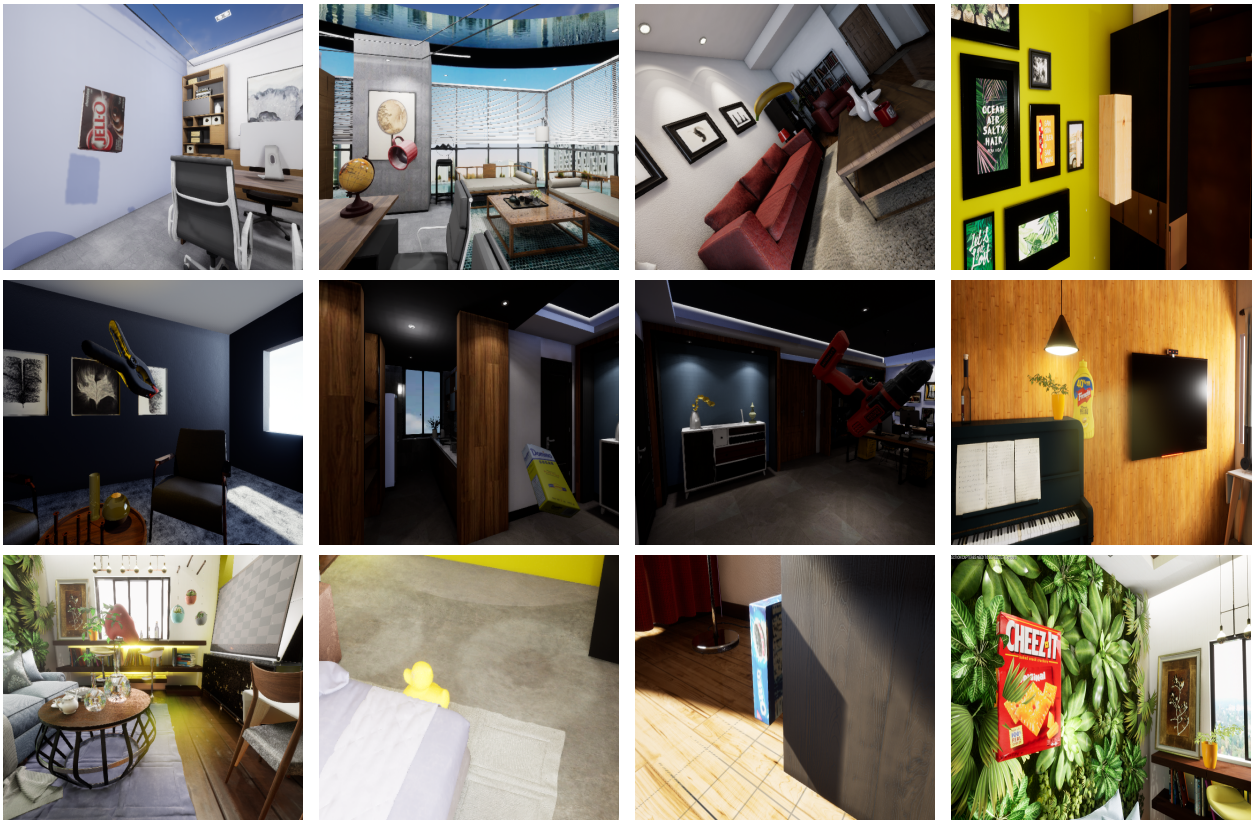


Figure 5.2. Examples of objects in different scene types.

Traditionally, such methods have used RGB-D images to infer the pose of the objects. The main drawback of such an approach is that depth cameras are not widely available (e.g., smartphones) and typically have low resolution and low frame rate, making it difficult to detect tiny, thin, or fast-moving objects. Therefore, RGB-only-based methods are preferred. Recently, many methods based on deep learning have been presented. These methods use convolutional neural networks to estimate the 6D pose of objects. Such neural networks estimate the pose by detecting keypoints [117], estimating a 3 dimensional bounding box [118]–[120], matching the input image with rendered images [58], [59], or directly treating pose estimation as a classification [121] or regression [122] problem.

With the increasing number of deep learning-based methods for RGB-only pose estimation, there is a need for more training data. Capturing real images is highly time-consuming. Therefore a faster approach is preferred. In addition, manually annotating the pose of objects is tedious and inaccurate. While several image synthesis methods have been presented

Figure 5.3. 3D object models in the YCB dataset.

Figure 5.4. 3D object models in the LINEMOD dataset.

[123] to generate new training samples automatically, the resulting images can lack realistic appearances. An efficient and effective alternative is photorealistic image rendering. Photorealistic rendering allows easy generation of a large number of images containing realistic lighting, occlusions, and real-world distortions with ground truth labeled automatically.

Many publicly available datasets consist of real-world images for 6D pose estimation. For example, T-LESS [124] is a dataset with 30 industrial objects that lack distinctive texture. There are 48.9K images in the T-LESS dataset. Many objects in the T-LESS dataset are symmetric, and their similarity is challenging for pose estimation tasks. The YCB dataset [125] contains 9.24K images of 77 real-life objects for benchmarking in robot grasping and manipulation tasks. The images in the YCB dataset are captured by the BigBIRD Object Scanning Rig and the Google scanner. The YCB-Video [122] dataset has 134K video frames for 21 household objects taken from the YCB dataset. The LINEMOD dataset [126] is another widely used public dataset for 6D pose estimation with various toys and household

objects. The LINEMOD OCCLUSION dataset [127], [128] is a complementary dataset for the LINEMOD dataset with 10K images under different lighting and occlusion conditions. The Rutgers APC [129] dataset includes real images of textured products used in the first Amazon Picking Challenge. The images in the Rutgers APC dataset with different poses and clutter conditions are mainly used for training algorithms in warehouse objects pick-and-place. The IC-MI dataset proposed in [130] has images for six objects heavily 2D and 3D cluttered with foreground occlusion.

Due to the large variety of datasets and evaluation metrics and the lack of a common benchmark procedure, the BOP dataset [131] was introduced. The BOP dataset has a thorough survey of 8 different datasets containing images and the evaluation methodologies. Additionally, the TUD Light and TOYOTA Light datasets are introduced in the BOP dataset. The TUD Light dataset includes images of three objects without occlusion under different illuminations. The TOYOTA Light dataset has 21 objects in total. Each object in TOYOTA Light is put on top of a table with different tablecloths and five different lighting conditions. The MVTec Industrial 3D Object Detection Dataset (MVTec ITODD) [132] contains 28 industrial objects. The dataset focuses more on practical and challenging tasks such as industrial bin picking and 3D object inspection. Besides datasets containing real captured images, some photorealistic rendered datasets, such as Falling Things (FAT) [133], have been made publicly available. The FAT dataset contains synthetic images with the 21 household object models from the YCB dataset.

In this work, we introduce a new dataset named Extra FAT. We follow a similar approach as in the FAT dataset [133], but we include a more significant number of object models and a larger variety of virtual scenes. This dataset includes rendered images containing many 3D object models from the most commonly used datasets for 6D pose estimation. Table 5.1 compares our dataset with previously presented datasets [1].

For each rendered image in the Extra FAT dataset, the location and rotation for both the virtual camera and the objects, and a registered pixel-level object segmentation mask with

[1]↑The number of objects in the BOP dataset is from the BOP benchmark paper [131]. There are more models provided for the BOP 2019 challenge.

Figure 5.5. 3D object models in the TYO-L [131], TUD-L [131], IC-MI [130], RU-APC [129], and T-LESS [124] datasets.

Table 5.1. Comparison of different 3D datasets: LINEMOD dataset (LM), YCB dataset (YCB), T-LESS (T-LESS), IC-MI dataset (IC-MI), TOYOTA Light dataset (TYO-L), Rutgers APC dataset (RU-APC), and TUD Light dataset (TUD-L)

Dataset	# obj	# frames	Type	LM	YCB	T-LESS	IC-MI	TYO-L	RU-APC	TUD-L
LINEMOD [126]	15	18K	real	✓						
LM OCC [127], [128]	15	18K	real	✓						
YCB-Video [122]	21	134K	real		✓					
FAT [133]	21	60K	rendered		✓					
T-LESS [124]	30	48.9K	real			✓				
IC-MI [130]	6	4.2K	real				✓			
TYO-L [131]	21	55.4K	combined					✓		
RU-APC [129]	24	10K	real						✓	
TUD-L [131]	6	62.3K	combined							✓
BOP [131]	89	>294K	combined	✓		✓	✓	✓	✓	✓
Extra FAT (ours)	110	825K	rendered	✓	✓	✓	✓	✓	✓	✓

640 × 480 resolution, shown in Figure 5.1, are provided. Such images and annotations can be used to train and test methods for object detection, segmentation, and pose estimation.

The images are simulated in five different indoor scenes with various illumination and occlusion conditions, as shown in Figure 5.2. The indoor scenes include everyday environments such as office spaces, living rooms, and kitchens. There are 825K images in total. The specifications for the Extra FAT dataset are shown in Table 5.3.

Table 5.2. Feature impact factor and time consumption

Feature	Histogram flatness	Color variability	Text edge count	Text color variance	Chroma around text	Chroma histogram flatness	White block ratio	Color block ratio
time(ms)	12.01	12.51	14.97	73.92	36.12	11.45	0.67	0.81
I_d	24.61%	13.84%	11.02%	1.9%	10.2%	3.4%	48.43%	13.65%

Table 5.3. Dataset Specification

Extra FAT Dataset	
Image Resolution	640 × 480
Field of view	90°
Number of frames	825K
Number of objects	110
Number of scenes	5

5.2 Dataset

5.2.1 Image Generation

The Extra FAT dataset is generated by rendering 110 3D object models: 21 household objects taken from the publicly available YCB dataset, 15 objects from the LINEMOD dataset, 30 objects from the T-LESS dataset, 14 objects from the Amazon Picking Challenge 2015 dataset, 6 objects from the IC-MI dataset, 3 objects from the TUD Light dataset and 21 objects from the TOYOTA Light dataset, as shown in Figure 5.3, Figure 5.4 and Figure 5.5.

As in the FAT dataset, we use the Unreal Engine 4 (UE4) [134], a commonly used tool for game development, to render 3D object models in the virtual game scenery. The open-source UnrealCV [134] plugin serves as a communication tool to generate photorealistic images and pose ground truth.

In the FAT dataset, objects are placed at random positions from where they fall. In the Extra FAT dataset, we move the object between pre-defined points (therefore, our objects are not technically falling but are flying).

We first manually specify some candidate points within the virtual scene. During the image generation process, pairs of candidate points are selected randomly and the virtual camera and object trajectories are defined by linear interpolation between the two points, as shown in Figure 5.6. While moving the objects between the pair of points, we apply a uniform random perturbation in the location and rotation of the object and the virtual camera.

Statistics of the objects in the Extra FAT dataset show that the distributions of the Yaw, Pitch, and Roll angles are uniform, indicating that the poses of objects in the dataset are comprehensive and representative for the general pose estimation task.

Figure 5.6. Linear interpolation trajectory from candidate location points.

In order to avoid placing the object out of the visible range of the camera, we constrain the relative location and rotation between the camera and objects. As shown in Figure 5.7, the pixel coordinates (p_x, p_y) obey the following constraint:

$$\begin{aligned} -\mu_x < p_x < \mu_x \\ -\mu_y < p_y < \mu_y \end{aligned} \tag{5.1}$$

where $\mu_x = 200$ and $\mu_y = 180$.

The constraint of the relative object position with respect to the camera (t_x, t_y, t_z) can be computed from the pixel coordinates:

$$\begin{aligned} t_x = p_x \frac{t_z}{f_x} \\ t_y = p_y \frac{t_z}{f_y} \end{aligned} \tag{5.2}$$

where f_x, f_y are the focal lengths in the x and y direction. The parameter t_z is in the range $(\theta_{z1}, \theta_{z2})$ to make sure that the object is in front of the camera and not too far from it, or too close to it. We set $\theta_{z1} = 0.3$ and $\theta_{z2} = 0.8$.

In order to avoid having objects highly occluded by a wall or other objects in the scenery, we add a constraint on the ratio of mask area to image size:

$$\frac{\sum_{mask} 1}{w \times h} > threshold \tag{5.3}$$

Images where the segmentation mask area to image size ratio is lower than $threshold = 0.05$ are discarded.

5.2.2 Training and Testing Setting

We propose three different training/testing split approaches. First, we provide a training/testing split with about 6,000 frames for training and 1,500 for testing for each object. Second, the training and testing sets can be divided by scene. Four scenes can be used for

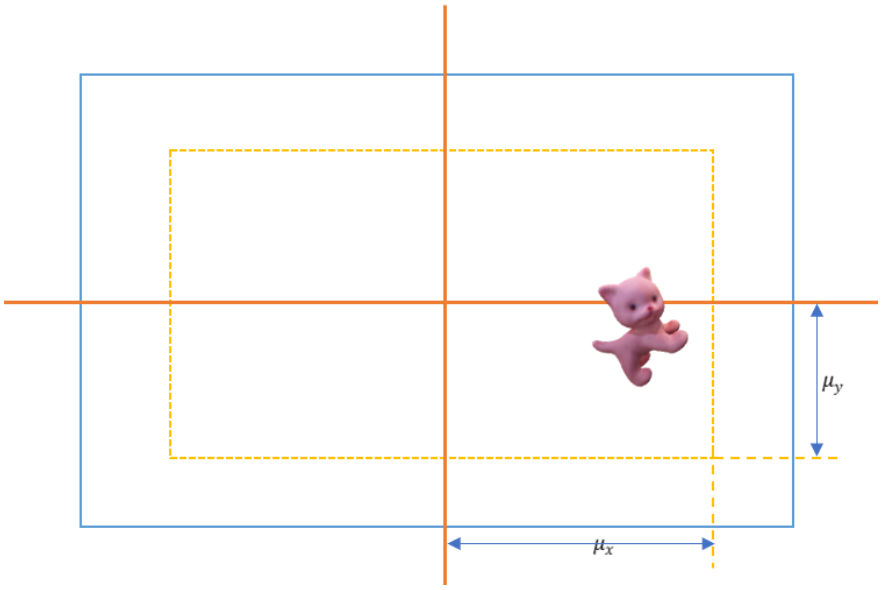


Figure 5.7. Pixel coordinate constraint.

training and the other one for testing. Finally, we propose using Extra FAT entirely as a training set and use the BOP [131] benchmark as a testing method.

5.3 Conclusion

This chapter presents a new dataset for 6D object pose estimation. By using photorealistic rendering, we obtain images with diversity in terms of illumination, reflection, and occlusion. These images can be used to train convolutional neural networks for object detection, segmentation, and pose estimation. We hope Extra FAT will help the community to propose novel algorithms for RGB-only image object segmentation and 6D pose estimation. We invite other researchers to generate new datasets, including objects from other commonly used datasets.

6. SUMMARY

This thesis introduces the algorithms to classify, detect, simulate and improve the quality of printed images with different defects.

First, we introduce image quality assessment, including defect classification, grading and detection. We implement a CNN for solving the defect classification problem, printed mottle defect grading and Faster R-CNN for the defect detection problem. The main contribution of this work is to propose a deep learning-based method instead of traditional feature-based methods for printed image quality assessment. For printed mottle defect grading, we propose a new deep learning-based method. Unlike traditional methods such as feature extraction using ΔE variation, our method utilizes a CNN for the first time to extract the feature automatically by stochastic gradient descent. Transfer learning and data augmentation methods are used to train a robust mottle defect grader. The proposed deep learning mottle characterization method can be used in mottle grading not only for the test image with the same uniform content as seen in the training set, but also for printed images with different contents. The mottle grading method achieves a 13.16% error rate in the T dataset with the same content and a 20.73% error rate in the combined dataset with different contents. The proposed method can also be generalized to other printed defects, such as streaks given an annotated streak dataset. The results prove the feasibility of deep learning methods for defects from different domains.

Second, we introduce Generative Adversarial Networks for printed image simulation. For deep learning-based approaches, a large amount of printed image data are required. Compared to other methods using real printed images as training data with traditional data augmentation, in this work, the defect images are automatically generated in our proposed simulation framework. Thus we can reduce the printing and scanning process in the data collection stage, which is costly and time-consuming. Generative Adversarial Network is an efficient tool to generate simulation data for training classification or detection algorithms. Thus, a deep CNN can be trained with a limited amount of scanned and digital image pairs. The qualitative results show that the simulated images could be used for training networks for printed image quality assessment.

Third, we focus on the printed image registration problem. This research investigates global and local image registration. For global image registration, we propose to use the deep learning framework PyTorch for the intensity-based image registration algorithm to reduce computation time. We create the SIMULATED dataset using data augmentation with elastic distortion, noise and color shift to simulate printed images. We also collect the HPLAB dataset with real digital and printed image pairs for image registration. For local deformable image registration, we implement the U-Net VoxelMorph-based method for printed images. Then we further propose the recurrent network-based method R-RegNet, which uses the weight-sharing feature network to extract features and reduces the model size. The weight sharing of GRU in our proposed recurrent network-based method R-RegNet could also speed up the iterative training process. We also introduce unsupervised loss function including similarity loss L_{sim}, smooth loss L_{smooth} and reconstruction loss L_{recon}. Experimental results prove the proposed R-RegNet method outperforms the U-Net VoxelMorph-based method in terms of mean square error or ground truth deformation error in all 3 test datasets.

Forth, a photorealistic image dataset simulation method is proposed for training deep neural networks. A new dataset with simulated images for 6D pose estimation, named Extra FAT, is introduced in this part. By using photorealistic rendering, the dataset has images with diversity in terms of illumination, reflection, and occlusion. The simulated images can also be used to train convolutional neural networks for object detection and segmentation.

REFERENCES

[1] M. Vans, S. Schein, C. Staelin, P. Kisilev, S. J. Simske, R. Dagan, and S. Harush, "Automatic visual inspection and defect detection on variable data prints," *Journal of Electronic Imaging*, vol. 20, no. 1, 2011.

[2] P. Kisilev and G. Ruckenstein, *Print defect detection*, US Patent 7,519,222, 2009.

[3] W. Yangping, X. Shaowei, Z. Zhengping, S. Yue, and Z. Zhenghai, "Real-time defect detection method for printed images based on grayscale and gradient differences," *Journal of Engineering Science & Technology Review*, vol. 11, no. 1, 2018.

[4] J. Lundström and A. Verikas, "Assessing print quality by machine in offset colour printing," *Knowledge-Based Systems*, vol. 37, pp. 70–79, 2013.

[5] N. Shankar, N. Ravi, and Z. Zhong, "On-line defect detection in web offset printing," in *the IEEE 4th International Conference on Control and Automation Proceedings*, 2003.

[6] J. Grice and J. P. Allebach, "The print quality toolkit: An integrated print quality assessment tool," *Journal of Imaging Science and Technology*, vol. 43, no. 2, 1999.

[7] W. Jang, M.-C. Chen, J. P. Allebach, and G. T.-C. Chiu, "Print quality test page," *Journal of Imaging Science and Technology*, vol. 48, no. 5, pp. 432–446, 2004.

[8] H. Santos-Villalobos, H. J. Park, C. Kim, P. Choe, R. Kumontoy, K. Low, K. Oldenburger, M. Ortiz, X. Lehto, M. Lehto, *et al.*, "A web-based self-diagnosis tool to solve print quality issues," in *NIP & Digital Fabrication Conference*, 2006.

[9] W. Jang and J. P. Allebach, "Simulation of print quality defects," *Journal of Imaging Science and Technology*, vol. 49, no. 1, pp. 1–18, 2005.

[10] Z. Pizlo, Y. Bang, and J. Allebach, "Banding assessment with controlled halftoning: The ten printer experiment," *Journal of Imaging Science and Technology*, vol. 50, no. 6, pp. 522–529, 2006.

[11] O. Arslan, Z. Pizlo, and J. Allebach, "Softcopy banding visibility assessment," *Journal of Imaging Science and Technology*, vol. 51, no. 3, pp. 271–281, 2007.

[12] B. Min, Z. Pizlo, and J. P. Allebach, "Development of softcopy environment for primary color banding visibility assessment," in *Image Quality and System Performance V, SPIE*, vol. 6808, 2008.

[13] T. H. Ha and J. P. Allebach, "Measurement and analysis of banding artifacts in color electrophotographic printers," in *Proceedings NIP24: IS&T's 24th International Conference on Digital Printing Technologies*, 2008.

[14] J. Zhang, S. Astling, R. Jessome, E. Maggard, T. Nelson, M. Shaw, and J. P. Allebach, "Assessment of presence of isolated periodic and aperiodic bands in laser electrophotographic printer output," in *Image Quality and System Performance X, SPIE*, vol. 8653, 2013.

[15] J. Zhang and J. P. Allebach, "Estimation of repetitive interval of periodic bands in laser electrophotographic printer output," in *Image Quality and System Performance XII, SPIE*, vol. 9396, 2015.

[16] W.-E. Huang, E. Maggard, R. Jessome, Y. Bang, M. Cho, and J. Allebach, "Cost-function-based repetitive interval estimation method with synthetic missing bands for periodic bands in electrophotographic printer," *Image Quality and System Performance XVI, IS&T Electronic Imaging*, no. 10, 2019.

[17] K. D. Donohue, C. Cui, and M. V. Venkatesh, "Wavelet analysis of print defects," in *IS&T's 2002 PICS Conference*, 2002.

[18] A. Eid, B. Cooper, and E. Rippetoe, "A unified framework for physical print quality," in *Image Quality and System Performance IV, SPIE-IS&T Electronic Imaging*, vol. 6494, 2007.

[19] X. Liu, G. Overall, T. Riggs, R. Silveston-Keith, J. Whitney, G. Chiu, and J. Allebach, "Wavelet-based figure of merit for macro-uniformity," in *Image Quality and System Performance X, SPIE*, vol. 8653, 2013.

[20] S. Hu, H. Nachlieli, D. Shaked, S. Shiffman, and J. P. Allebach, "Color-dependent banding characterization and simulation on natural document images," in *Color Imaging XVII: Displaying, Processing, Hardcopy, and Applications, SPIE*, vol. 8292, 2012.

[21] X. Jing, H. Nachlieli, D. Shaked, S. Shiffman, and J. P. Allebach, "Masking mediated print defect visibility predictor," in *Image Quality and System Performance IX, SPIE*, vol. 8293, 2012.

[22] J. Zhang, H. Nachlieli, D. Shaked, S. Shiffman, and J. P. Allebach, "Psychophysical evaluation of banding visibility in the presence of print content," in *Image Quality and System Performance IX, SPIE*, vol. 8293, 2012.

[23] Y. Liu and J. P. Allebach, "A computational texture masking model for natural images based on adjacent visual channel inhibition," in *Image Quality and System Performance XI, SPIE*, vol. 9016, 2014.

[24] Y. Liu and J. P. Allebach, "A patch-based cross masking model for natural images with detail loss and additive defects," in *Human Vision and Electronic Imaging XX, SPIE*, vol. 9394, 2015.

[25] Z. Xiao, S. Xu, E. Maggard, K. Morse, M. Shaw, and J. Allebach, "Detection of color fading in printed customer content," *Color Imaging XXIII: Displaying, Processing, Hardcopy, and Applications, IS&T Electronic Imaging*, no. 16, 2018.

[26] D. Wolin, "Enhanced mottle measurement," in *IS&T's 2002 PICS Conference*, 2002.

[27] P.-A. Johansson, "Print mottle evaluation by band-pass image analysis," *Advances in Printing Science and Technology*, vol. 22, 1994.

[28] R. R. Rosenberger, "Stochastic frequency distribution analysis as applied to ink jet print mottle measurement," in *Proceeding of IS&T's 17th Non-Impact Printing Conference*, 2001.

[29] C.-M. Fahlcrantz, P.-Å. Johansson, and P. Åslund, "The influence of mean reflectance on perceived print mottle," *Journal of imaging Science and Technology*, vol. 47, no. 1, pp. 54–59, 2003.

[30] A. H. Eid, B. E. Cooper, and E. E. Rippetoe, "Characterization of mottle and low-frequency print defects," in *Image Quality and System Performance V, SPIE*, vol. 6808, 2008.

[31] S. Khullar, E. Saber, S. Dianat, J. Trask, R. Lawton, and M. Shaw, "Automatic multi-resolution spatio-frequency analysis for print mottle evaluation," in *Color and Imaging Conference*, 2009.

[32] M. Q. Nguyen, R. Jessome, S. Astling, E. Maggard, T. Nelson, M. Shaw, and J. P. Allebach, "Perceptual metrics and visualization tools for evaluation of page uniformity," in *Image Quality and System Performance XI, SPIE*, vol. 9016, 2014.

[33] N. Yan, E. Maggard, R. Fothergill, R. J. Jessome, and J. P. Allebach, "Autonomous detection of ISO fade point with color laser printers," in *Image Quality and System Performance XII*, vol. 9396, Jan. 2015.

[34] Y. Ju, E. Maggard, R. Jessome, and J. P. Allebach, "Autonomous detection of text fade point with color laser printers," in *Image Quality and System Performance XII, SPIE*, vol. 9396, 2015.

[35] X. Jing, S. Astling, R. Jessome, E. Maggard, T. Nelson, M. Shaw, and J. P. Allebach, "Electrophotographic ghosting detection and evaluation," in *Proceedings of NIP-31 IS&T's 2015 Conference on Digital Fabrication and Digital Printing*, 2015.

[36] J. Wang, T. Nelson, R. Jessome, S. Astling, E. Maggard, M. Shaw, and J. P. Allebach, "Local defect detection and print quality assessment," *Image Quality and System Performance XIII, IS&T Electronic Imaging*, no. 13, pp. 1–7, 2016.

[37] Q. Chen, R. Jessome, E. Maggard, and J. P. Allebach, "Segmentation-based detection of local defects on printed pages," *Image Quality and System Performance XVI, IS&T Electronic Imaging*, no. 10, pp. 301–1, 2019.

[38] X. Xiang, R. Jessome, E. Maggard, Y. Bang, M. Cho, and J. P. Allebach, "Blockwise based detection of local defects," *Image Quality and System Performance XVI, IS&T Electronic Imaging*, no. 10, pp. 303–1, 2019.

[39] H. S. Rosario, E. Saber, W. Wu, and K. Chandu, "Streak detection in mottled and noisy images," *Journal of Electronic Imaging*, vol. 16, no. 4, p. 043 005, 2007.

[40] R. Zhang, E. Maggard, R. Jessome, Y. Bang, M. Cho, and J. P. Allebach, "Block window method with logistic regression algorithm for streak detection," *Image Quality and System Performance XVI, IS&T Electronic Imaging*, no. 10, pp. 300–1, 2019.

[41] B. Streckel, B. Steuernagel, E. Falkenhagen, E. Jung, *et al.*, "Objective print quality measurements using a scanner and a digital camera," in *IS&T International Conference on Digital Production Printing and Industrial Applications*, vol. 5, 2003.

[42] O. A. Ugbeme, E. Saber, and W. Wu, "An automated defect classification algorithm for printed documents," *Proceedings of International Congress of Imaging Science*, pp. 317–320, 2006.

[43] O. A. Ugbeme, E. Saber, W. Wu, and K. Chandu, "Automated algorithm for the identification of artifacts in mottled and noisy images," *Journal of Electronic Imaging*, vol. 16, no. 3, p. 033 015, 2007.

[44] K. Chandu, E. Saber, and W. Wu, "A mutual information based automatic registration and analysis algorithm for defect identification in printed documents," in *2007 IEEE International Conference on Image Processing*, vol. 3, 2007.

[45] M. Q. Nguyen and J. P. Allebach, "Controlling misses and false alarms in a machine learning framework for predicting uniformity of printed pages," in *Image Quality and System Performance XII, SPIE*, vol. 9396, 2015.

[46] W. Wang, G. Overall, T. Riggs, R. Silveston-Keith, J. Whitney, G. Chiu, and J. P. Allebach, "Figure of merit for macrouniformity based on image quality ruler evaluation and machine learning framework," in *Image Quality and System Performance X, SPIE*, vol. 8653, 2013.

[47] W. Wang, J. P. Allebach, and Y. Guo, "Image quality evaluation using image quality ruler and graphical model," in *IEEE International Conference on Image Processing*, 2015.

[48] Y. Yang, U. Sarkar, I. Borrell, and J. P. Allebach, "Inkjet quality ruler experiments and print uniformity predictor," *Image Quality and System Performance XVII, IS&T Electronic Imaging*, no. 9, pp. 373–1, 2020.

[49] R. Zhang, Y. Yang, E. Maggard, Y. Bang, M. Cho, and J. P. Allebach, "A comprehensive system for analyzing the presence of print quality defects," *Image Quality and System Performance XVII, IS&T Electronic Imaging*, no. 9, pp. 314–1, 2020.

[50] R. Zhang, E. Maggard, Y. Bang, M. Cho, and J. P. Allebach, "Region of interest extraction for image quality assessment," *Image Quality and System Performance XVII, IS&T Electronic Imaging*, no. 9, pp. 321–1, 2020.

[51] J. C. Briggs, D. J. Forrest, A. H. Klein, and M.-K. Tse, "Living with ISO-13660: Pleasures and perils," in *NIP & Digital Fabrication Conference*, 1999.

[52] ISO/IEC JTC 1, Information technology, Subcommittee SC 28, Office equipment, "Information technology – Office equipment – Measurement of image quality attributes for hardcopy output – Monochrome text and graphic images," en, International Organization for Standardization, Standard ISO/IEC 24790:2017.

[53] ISO/IEC JTC 1, Information technology, Subcommittee SC 28, Office equipment, "Information technology – Office equipment – Measurement of image quality attributes for hardcopy output – Binary monochrome text and graphic images," en, International Organization for Standardization, Standard ISO/IEC 13660:2001.

[54] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[55] W. Xi, J. Chen, Q. Lin, and J. P. Allebach, "High-accuracy automatic person segmentation with novel spatial saliency map," in *IEEE International Conference on Image Processing*, 2019.

[56] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2015.

[57] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proceedings of the European Conference on Computer Vision*, 2016.

[58] D. M. Montserrat, J. Chen, Q. Lin, J. P. Allebach, and E. J. Delp, "Multi-View matching network for 6D pose estimation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[59] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep iterative matching for 6D pose estimation," *Proceedings of the European Conference on Computer Vision*, pp. 683–698, 2018.

[60] *Streak*. [Online]. Available: http://printwiki.org/Streak.

[61] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Proceedings of the Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105, 2012.

[64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[65] I. Goodfellow, Y. Bengio, and A. Courville, "Introduction," in *Deep Learning*, vol. 1, Cambridge, MA: MIT Press, 2016, p. 20.

[66] L. N. Smith, "Cyclical learning rates for training neural networks," in *IEEE Winter Conference on Applications of Computer Vision*, 2017.

[67] F.-C. Chen and M. R. Jahanshahi, "NB-CNN: Deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4392–4400, 2017.

[68] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, 2017.

[69] Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 731–747, 2018.

[70] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *Proceedings of the European Conference on Computer Vision*, 2014.

[71] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard, "Adaptive data augmentation for image classification," in *IEEE International Conference on Image Processing*, 2016, pp. 3688–3692.

[72] R. W. Keener, *Theoretical statistics: Topics for a core course*. Springer, 2011.

[73] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[74] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[75] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[76] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with PixelCNN decoders," in *Proceedings of the Advances in Neural Information Processing Systems*, 2016.

[77] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the Advances in Neural Information Processing Systems*, 2014.

[78] W. R. Tan, C. S. Chan, H. E. Aguirre, and K. Tanaka, "ArtGAN: Artwork synthesis with conditional categorical GANs," in *IEEE International Conference on Image Processing*, 2017.

[79] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *European Conference on Computer Vision Workshops*, 2018.

[80] Y. Cao, Z. Zhou, W. Zhang, and Y. Yu, "Unsupervised diverse colorization via generative adversarial networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2017.

[81] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," *Proceedings of the Advances in Neural Information Processing Systems*, 2017.

[82] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1125–1134.

[83] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Proceedings of the European Conference on Computer Vision*, Springer, 2016, pp. 649–666.

[84] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[85] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv preprint arXiv:1612.02136*, 2016.

[86] Z. Xiao, M. Nguyen, E. Maggard, M. Shaw, J. P. Allebach, and A. Reibman, "Real-time print quality diagnostics," *Electronic Imaging, Image Quality and System Performance XIV*, no. 12, pp. 174–179, 2017.

[87] B. Zitova and J. Flusser, "Image registration methods: A survey," *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[88] G. Haskins, U. Kruger, and P. Yan, "Deep learning in medical image registration: A survey," *Machine Vision and Applications*, vol. 31, no. 1, pp. 1–18, 2020.

[89] Y. Fu, Y. Lei, T. Wang, W. J. Curran, T. Liu, and X. Yang, "Deep learning in medical image registration: A review," *Physics in Medicine & Biology*, vol. 65, no. 20, 20TR01, 2020.

[90] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the Association for Computing Machinery*, vol. 24, no. 6, pp. 381–395, 1981.

[91] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[92] Z. Luo, T. Shen, L. Zhou, S. Zhu, R. Zhang, Y. Yao, T. Fang, and L. Quan, "GeoDesc: Learning local descriptors by integrating geometry constraints," in *Proceedings of the European Conference on Computer Vision*, 2018.

[93] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[94] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[95] *PyTorch*, URL: http://pytorch.org.

[96] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Proceedings of the Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[97] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.

[98] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[99] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the International Conference on Learning Representations*, 2014.

[100] M. Simonovsky, B. Gutiérrez-Becker, D. Mateus, N. Navab, and N. Komodakis, "A deep metric for multimodal registration," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2016.

[101] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca, "Voxelmorph: A learning framework for deformable medical image registration," *IEEE Transactions on Medical Imaging*, vol. 38, no. 8, pp. 1788–1800, 2019.

[102] S. Zhao, T. Lau, J. Luo, I. Eric, C. Chang, and Y. Xu, "Unsupervised 3D end-to-end medical image registration with volume tweening network," *IEEE journal of biomedical and health informatics*, vol. 24, no. 5, pp. 1394–1404, 2019.

[103] S. Zhao, Y. Dong, E. I. Chang, and Y. Xu, "Recursive cascaded networks for unsupervised medical image registration," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[104] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010.

[105] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[106] C. Hernandez-Matas, X. Zabulis, A. Triantafyllou, P. Anyfanti, S. Douma, and A. A. Argyros, "FIRE: Fundus image registration dataset," *Modeling and Artificial Intelligence in Ophthalmology*, vol. 1, no. 4, 2017.

[107] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European conference on computer vision*, Springer, 2020.

[108] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2758–2766, 2015.

[109] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[110] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[111] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013.

[112] B. Zhao, X. Li, and X. Lu, "Cam-rnn: Co-attention model based rnn for video captioning," *IEEE Transactions on Image Processing*, vol. 28, no. 11, 2019.

[113] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, IEEE, 2015.

[114] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv preprint arXiv:1704.02971*, 2017.

[115] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[116] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[117] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-DoF object pose from semantic keypoints," *Proceedings of the International Conference on Robotics and Automation*, pp. 2011–2018, 2017.

[118] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," *Proceedings of the Conference on Robot Learning*, 2018.

[119] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1530–1538, 2017.

[120] M. Rad and V. Lepetit, "BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth," *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[121] A. Kundu, Y. Li, and J. M. Rehg, "3D-RCNN: Instance-level 3D object reconstruction via render-and-compare," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[122] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," *Proceedings of Robotics: Science and Systems*, 2018.

[123] D. Mas Montserrat, Q. Lin, J. Allebach, and E. J. Delp, "Logo detection and recognition with synthetic images," *Proceedings of the IS&T International Symposium on Electronic Imaging*, 2018.

[124] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2017.

[125] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[126] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," *Proceedings of the Asian Conference on Computer Vision*, pp. 548–562, 2012.

[127] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," *Proceedings of the European Conference on Computer Vision*, 2014.

[128] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 954–962, 2015.

[129] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1179–1185, 2016.

[130] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," *Proceedings of the European Conference on Computer Vision*, pp. 462–477, 2014.

[131] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, *et al.*, "BOP: Benchmark for 6D object pose estimation," *Proceedings of the European Conference on Computer Vision*, 2018.

[132] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger, "Introducing MVTec ITODD-a dataset for 3D object recognition in industry," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2200–2208, 2017.

[133] J. Tremblay, T. To, and S. Birchfield, "Falling things: A synthetic dataset for 3D object detection and pose estimation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[134] W. Qiu and A. Yuille, "UnrealCV: Connecting computer vision to unreal engine," *Proceedings of the European Conference on Computer Vision*, pp. 909–916, 2016.

VITA

Jianhang Chen is a Ph.D. candidate from the School of Electrical and Computer Engineering, Purdue University. Before that, he joined NTU Robotics Lab, and received his M.S. degree in Mechanical Engineering from National Taiwan University in 2017. He received his B.S. degree in Information and Computational Science from Beihang University in 2014. His research focuses on computer vision and deep learning, with emphasis on printed image quality and deep learning. He also regularly reviews for IEEE, SPIE sponsored conferences and journals. He will be a software engineer at Google after Ph.D. graduation. Before that, he was in Amazon Alexa as an applied scientist intern, and Qualcomm as an engineering intern. During his study in Purdue, he collaborated with HP Labs in his research.