Skolkovo Institute of Science and Technology

LEARNABLE WARPING-BASED APPROACH TO IMAGE RE-SYNTHESIS WITH APPLICATION TO GAZE REDIRECTION

Doctoral Thesis

by

DANIIL KONONENKO

DOCTORAL PROGRAM IN
COMPUTATIONAL AND DATA SCIENCE
AND ENGINEERING

Supervisor
Associate Professor Victor Lempitsky

Moscow - 2017

Abstract

The thesis proposes an approach to the image re-synthesis problem where the target transformation is described by a dataset of pairs of input images and corresponding outputs. The approach is based on a warping field concept: pixels of the output image are sampled from the input. The warping field predictor is learned from a dataset of examples. The new approach is applicable to image re-synthesis problems, where the transformations can be well approximated by warping, in particular where dis-occlusion and global color changes are minimal. The suggested methods are illustrated and evaluated on a gaze redirection task. Such learning-based re-synthesis can achieve convincing gaze redirection based on monocular input and that the learned systems generalize well to people and imaging conditions unseen during training.

Four methods of learning a warping field predictor are described and compared. The first system is based on efficient decision forest predictors and redirects the gaze by a fixed angle in real time (on a single CPU), which is particularly suitable for the videoconferencing gaze correction. The second system is based on a deep architecture, allowing gaze redirection by a range of angles. The second system achieves higher photorealism, while being several times slower. The third system is based on real-time decision forests at test time, while using the supervision from a teacher deep network during training. The third system approaches the quality of the teacher network in our experiments and thus provides a highly realistic real-time monocular solution to the gaze correction problem. The fourth system is based on a pair of deep networks where the first network maps eye images to a latent space, whereas the second maps pairs of latent representations to warping fields implementing the transformation between the pair of original images. Both networks are trained in an unsupervised manner, while the gaze-annotated images are only used to estimate displacements in the latent space that are characteristic of certain gaze redirections. In-depth assessment and comparisons of the proposed systems based on quantitative measurements and a user study is presented.

Acknowledgements

I wish to express my very great appreciation to my advisor Victor Lempitsky for his eagerness to discuss and share ideas and knowledge, enthusiastic encouragement, useful critiques of this research work and patience with his students. Here at Skoltech, Victor created a unique Computer Vision group where only desire and diligence are required from a student, with everything else on the path to great academic achievements taken care of.

A very special recognition should be given to Yaroslav Ganin and Diana Sungatullina for their invaluable contribution to the gaze redirection project. You are fabulous collaborators and I enjoyed working with you very much. I am also thankful to Nikita Klyuchnikov for helping with data collection and producing a terrific head stand design. Similarly, my heartfelt thanks to Sergey Ulyakhin and Igor Seleznev for helping with the commercialization. I thank Eric Sommerlade and his team at RealD for the interest shown in our work, wishing them every success in developing the project and achieving all the goals.

My special thanks are also extended to all the members of Skoltechs Computer Vision group, Andrey Kuzmin, Vadim Lebedev, Evgeniya Ustinova, Dmitry Ulyanov, Andrey Shadrikov, Alexander Vakhitov and Victor Kulikov. Over these years, you have been a wonderful company and an inexhaustible source of ingenious and viable ideas. I am particularly grateful to CDISE PhD committee members, Maxim Fedorov, Evgeny Burnaev, Ivan Oseledets, Stamatis Lefkimmiatis and Alexander Bernstein for their helpful comments and advice on how to write a thesis and present the results. My sincere gratitude also goes to CDISE and Skoltech community for creating such a wonderful learning and working environment. I wish to thank Skoltech football team for making every Monday evening special and keeping me in shape.

Finally, I extend my very special thanks to my family. I wish to thank my parents Sergey and Irina Kononenko for instilling a passion for science in me. Although we are thousands of miles apart, deep inside I feel a strong connection with you and with my brother Ilya, my grandmother Natalya and my uncle Mikhail. I thank my beloved wife Ekaterina who remained an endless source of inspiration for me throughout the process of this thesis and beyond.

Contents

Abstract i

Acknowledgements ii

List of Figures vi

List of Tables viii

1 Introduction 1
- 1.1 Image re-synthesis . . . 1
  - 1.1.1 Related work on image re-synthesis via regression of target image . 4
    - 1.1.1.1 Variational Autoencoders . . . 4
    - 1.1.1.2 Generative Adversarial Networks . . . 5
  - 1.1.2 Related work on image re-synthesis via warping . . . 9
- 1.2 Gaze redirection . . . 12
  - 1.2.1 Previous work on gaze redirection . . . 15
    - 1.2.1.1 Gaze correction via 3D view synthesizing . . . 17
    - 1.2.1.2 Gaze correction via eye replacement . . . 19
  - 1.2.2 Recent work on gaze redirection . . . 21
- 1.3 Contributions and scope . . . 21
- 1.4 Work overview . . . 25
  - 1.4.1 Contributions and scientific novelty . . . 25
  - 1.4.2 Academic and non-academic impact . . . 26

2 Machine-learning-based image re-synthesis 29
- 2.1 Image re-synthesis by pixel-wise replacement . . . 29
  - 2.1.1 General pipeline for image re-synthesis . . . 33
- 2.2 Gaze redirection by re-synthesis of eyes vicinity . . . 34
- 2.3 Database collection and eye localization . . . 35
  - 2.3.1 Related work on Facial Alignment . . . 39
- 2.4 General system for gaze redirection . . . 41
  - 2.4.1 Assessment methods . . . 42

3 Weakly-supervised random forest for image re-synthesis 44
- 3.1 Overview of Random Forests . . . 44

3.1.1 Training a Decision Tree . . . 45
3.1.2 Random Forests . . . 47
3.1.3 Application of Random Forests in Computer Vision . . . 48
3.2 Warping flow forest . . . 50
3.2.1 Image-independent warping field . . . 50
3.2.2 Architecture of warping flow forest . . . 51
3.2.3 Learning . . . 52
3.2.4 Implementation details . . . 54
3.3 Experiments . . . 56

4 Image re-synthesis using deep warping architecture 63
4.1 Overview of Deep Learning . . . 63
4.2 Coarse-to-fine warping . . . 66
4.3 Input embedding . . . 68
4.4 Lightness correction module . . . 69
4.5 Training procedure . . . 71
4.6 Experiments . . . 71
4.6.1 Quantitative evaluation . . . 71
4.6.1.1 Models. . . . 71
4.6.1.2 15° correction. . . . 72
4.6.1.3 Arbitrary vertical redirection. . . . 73
4.6.2 Perceptual quality . . . 73
4.6.2.1 User study . . . 74
4.6.2.2 Continuous gaze redirection. . . . 76
4.6.3 Incorporating registration. . . . 76

5 Regression random forest using neural network supervision 79
5.1 Related work on teacher-student architectures . . . 79
5.2 Learning . . . 81
5.3 Experiments . . . 83
5.3.1 Evaluation using Mean Squared Error . . . 83
5.3.2 Was the gaze actually redirected? . . . 84
5.3.3 Qualitative evaluation . . . 86
5.3.4 User study . . . 90
5.3.5 Computational speed and memory demands . . . 93

6 Semi-supervised gaze redirection using deep embedding learning 95
6.1 Related work . . . 96
6.2 Unsupervised training of gaze redirection . . . 99
6.3 Semi-supervised learning and analogies . . . 101
6.4 Experiments . . . 103

7 Conclusions and summary 109
7.1 Discussion . . . 110

Bibliography **113**

List of Figures

1.1 Facial feature interpolation [1] . . . 2
1.2 Video interpolation [2] . . . 2
1.3 Image super-resolution [3] . . . 2
1.4 Examples of re-synthesis problem . . . 2
1.5 Unpooling layer [4] . . . 4
1.6 Examples of generated images using VAEs with 2D codes [5] . . . 5
1.7 Image editing on the natural image manifold [6] . . . 7
1.8 Photo editing in Neural Photo Editor [7] . . . 8
1.9 Reconstruction produced by autoencoder with ℓ_2 and GAN losses . . . 10
1.10 Multi-view network architecture [8] . . . 11
1.11 Vertical gap between camera and conference window . . . 12
1.12 Gaze correction in videoconferencing: transformation examples . . . 13
1.13 Gaze redirection for image and video post-processing [9] . . . 14
1.14 Architecture of the 3D teleconferencing system [10] . . . 16
1.15 Gaze correction workflow [11] . . . 18
1.16 Eye model from [12] . . . 20
1.17 Eye-editing approach [13] . . . 22

2.1 Visualization of bilinear interpolation [14] . . . 31
2.2 Comparison of warping and non-warping approaches, training set . . . 32
2.3 Comparison of warping and non-warping approaches, validation set . . . 33
2.4 Examples of monocular gaze redirection results . . . 35
2.5 Dataset collection process . . . 36
2.6 Examples from the dataset . . . 37
2.7 Eye localization . . . 38
2.8 Examples of training data . . . 39
2.9 Landmark estimates at different levels of the cascade [15] . . . 40

3.1 Body part classification [16] . . . 49
3.2 Architecture of warping flow tree . . . 51
3.3 Error plot of warping flow forests . . . 57
3.4 Visualization of results of warping flow forest on Columbia Gaze dataset . . . 58
3.5 Visualization of results of warping flow forest on Skoltech dataset . . . 60
3.6 Failure cases of warping flow forest . . . 61
3.7 Visualization of results of warping flow forest on face images . . . 61

4.1 Deep warp overview . . . 67
4.2 Deep warp architecture . . . 67

4.3 Lightness Correction Module examples . . . 69
4.4 Architecture of Lightness Correction Module . . . 70
4.5 Quantitative comparison of deep warp versus weakly-supervised random forest . . . 73
4.6 Distribution of errors over different correction angles . . . 74
4.7 Qualitative comparison of deep warp versus weakly-supervised random forest . . . 75
4.8 Vertical continuous gaze redirection . . . 77
4.9 Horizontal continuous gaze redirection . . . 78

5.1 Examples of warping field . . . 81
5.2 Quantitative evaluation of nn-supervised random forest . . . 84
5.3 The assessment of the redirection angles . . . 87
5.4 Qualitative evaluation of nn-supervised random forest . . . 88
5.5 Examples of warping field . . . 89
5.6 Inherent dimensionality of the warping fields manifold . . . 89
5.7 Results on multiple resolutions . . . 90
5.8 Comparison with monocular gaze correction method from [17] . . . 91
5.9 Screen-shot from user study . . . 93
5.10 Results of user study for pairs of methods . . . 94

6.1 The concept of visual analogy making [18] . . . 97
6.2 Manipulation rotation results from [19] . . . 99
6.3 Architecture of unsupervised gaze redirection training . . . 100
6.4 Image analogy-making in test-time . . . 103
6.5 Demonstration of analogy property of the learned embedding . . . 104
6.6 Quantitative evaluation of semi-supervised learning . . . 106
6.7 Quantitative evaluation of semi-supervised learning . . . 107
6.8 Continuous vertical redirection, semi-supervised method . . . 108

List of Tables

1.1 Comparison of related gaze redirection works' characteristics 16
1.2 Comparative analysis of gaze redirection systems 24

3.1 Error table of warping flow forests 59

4.1 User study on images of eyes 76

5.1 User study on images of faces 92

Chapter 1

Introduction

1.1 Image re-synthesis

Image re-synthesis is about transforming the input image in a certain way. The examples of applying of image re-synthesis in computer vision include, but are not limited to changing facial features (Figure 1.1), video interpolation (Figure 1.2), image super-resolution (Figure 1.3) and gaze redirection (Section 1.2).

The image re-synthesis task is related to both computer graphics and computer vision. In computer graphics, image synthesis, or rendering, is a process of generating a photo-realistic image from a physical model [20]. All attributes of the scene are given in this model. In computer vision, on the contrary, the image is given and the physical model is unknown. Understanding the scene is thus one of the main problems of computer vision, with the different tasks of recognition, segmentation, 3D reconstruction and others, remaining unsolved in their general form.

Recovering the physical model using computer vision and then obtaining its photo-realistic rendering using computer graphics is one way to handle the image re-synthesis problem. For example, [13] uses physically-based face rendering and [9] a physical model of the eye to solve the gaze redirection problem (Figure 1.13). However, a model that meets the requirements of rendering is not always possible to recover. Moreover, such a complex approach would not work if either the computer vision model is not accurate enough or the rendering step fails. Finally, building complex multi-parametric physical models is often slow.

An alternative way to define the target transformation is through a database of transformation examples. They may vary from one database to another, featuring different images with and without a certain attribute to be changed, for instance, smiling and

FIGURE 1.1: Adding different attributes to the same person. Top: older, mouth open, eyes open. Bottom: smiling, facial hair, glasses. Figure taken from [1].

FIGURE 1.2: Video interpolation on the KITTI dataset. The middle frame is predicted from left and right frames. Figure taken from [2].

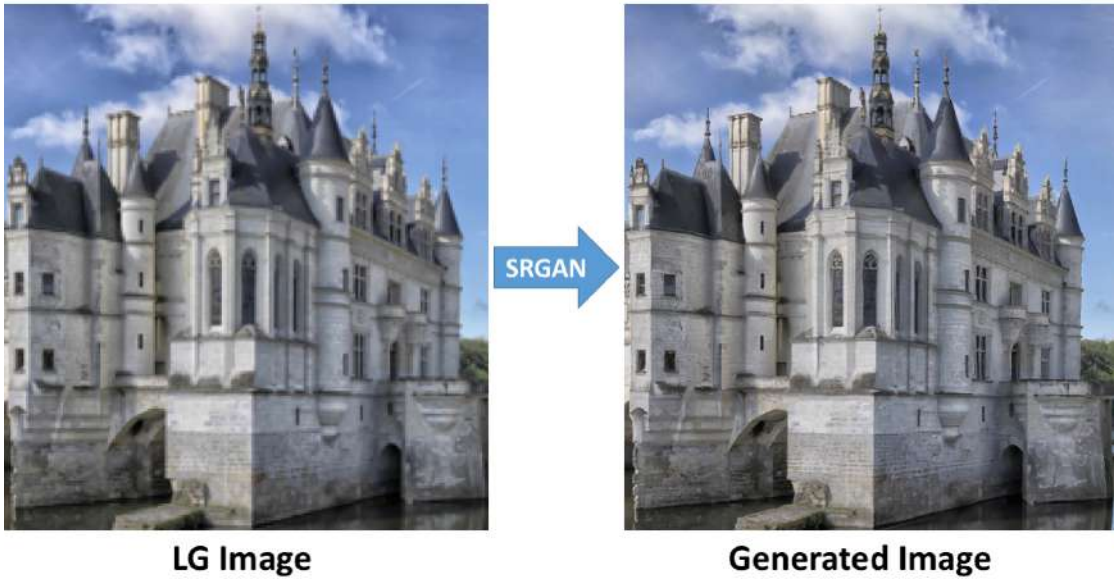


FIGURE 1.3: Image super-resolution. The resolution of the left image is enhanced on the fight one. The method utilizes the Generative Adversarial Networks approach (Section 1.1.1). Figure taken from [3].

FIGURE 1.4: Examples of re-synthesis problem.

non-smiling faces from the CelebA dataset [21]. In some situations, aligned sets of examples of a different type can be obtained if ground-truth images are provided. In [2], for example, two input frames and the middle ground-truth image are picked from the UCF101 dataset [22]. This alternative way is an instance of the machine learning approach where a model does not consist of a fixed set of attributes, as opposed to a physical model, but is relatively free to learn any dependencies in the training data. While even a complex physical model is always a simplification of the real world, the machine learning model, if perfectly trained on a large training dataset, has the potential to model all the necessary data variations in more precise manner and ensure higher photorealism.

Several approaches to learning-based image re-synthesis are getting increasingly popular, with neural-network-based image synthesis receiving growing attention [4, 23–29]. More closely related to this work are the methods that learn to transform in. These methods proceed by learning internal compact representations of images using encoder-decoder (autoencoder) architectures, and then transforming the images by changing their internal representation in a certain way that can be trained from examples (for a more detailed review of [18, 19] see Section 6.1). This approach can be combined with several ideas that have been reported to improve the result (convolutional and up-convolutional layers [4, 30], adversarial loss [24], variational autoencoders [5], perceptual loss [29]). The recent paper [28] addresses the problem using a new cross-convolutional layer that models both correlation between motion and image content and uncertainty of the motion field. The network then encodes the input image pyramid into multiple feature maps and convolves these maps with different kernels. As discussed in more detail below, the existing approaches to re-synthesis, such as those based on auto-encoders and adversarial networks, often fail to produce fully realistic images at high resolution.

In my work, I concentrate on the scenario, when the target transformation is defined by providing the pairs of inputs and corresponding outputs. To overcome the problems of existing approaches with generating photorealistic images, I suggest the warping field based method. The idea of the method is that pixels of the output image are sampled from the input image in places, defined by the warping field. The learning task is therefore reduces to learning the warping field predictor from the training pairs of images. As there is no ground truth warping fields in the dataset, this is an instance of a weakly-supervised problem. I illustrate and evaluate suggested methods for predicting the warping field on the gaze redirection task. Before introducing further details on the proposed approach, I discuss the related works in recent literature.

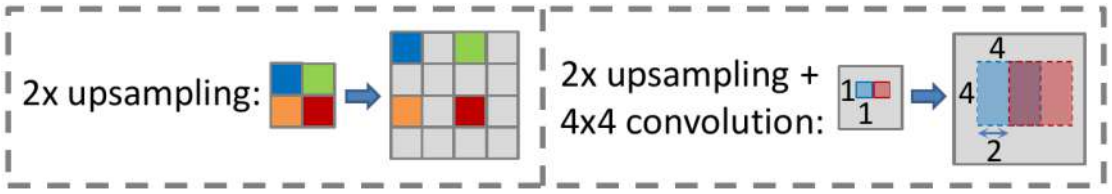


FIGURE 1.5: Illustration of upsampling (left) and upsampling+convolution (right) as used in the generative network. Figure taken from [4].

1.1.1 Related work on image re-synthesis via regression of target image

Generally, the class of image synthesis problems under review is related to manifold learning algorithms. It is hard to learn functions with interesting variations in high-dimensional image space. Manifold learning algorithms suggest that most of the space consists of invalid inputs and that interesting inputs appear along a collection of manifolds only.

The class of manifold learning models at hand is generative models that transform samples of latent variables z to target samples x or to the parametric distribution over x using a differentiable function $g(z)$ which is usually represented by a neural network. Typically, generative models are learned in an unsupervised scenario, with only a dataset of real samples x given. The paper [4], however, considers a case where the correspondence between z and x is given. Computer-rendered images of chairs are used as training data, and z denotes the parameters given to the rendering engine (type of chair, viewpoint and distance to the chair). The network architecture is a deep convolutional network with unpooling layers (Figure 1.5). Each pixel in the feature map is replaced by a patch with one non-zero value. Followed by convolution, this operation could be regarded as the opposite of convolution+pooling in the standard CNN. A combination of the segmented-out chair image loss reconstruction and the segmentation mask as an objective is used for training a generative network.

1.1.1.1 Variational Autoencoders

An interesting example of manifold learning and generative models is Variational Autoencoders [5, 31] (VAEs) that use a latent representation space $\mathbb{Z}$ too. The sample x is also generated by a generator network $g(z)$, where $z \in \mathbb{Z}$ is a sample from a distribution $p_{model}(z)$ in the latent space. Thus, x is sampled for a distribution $p_{model}(x; g(z)) = p_{model}(x|z)$. During training on an unlabeled dataset with examples $x \in \mathbb{R}^N$, the encoder network $q(z|x)$ is used to map the inputs to the latent space $\mathbb{Z}$, and the generator is viewed as a decoder.

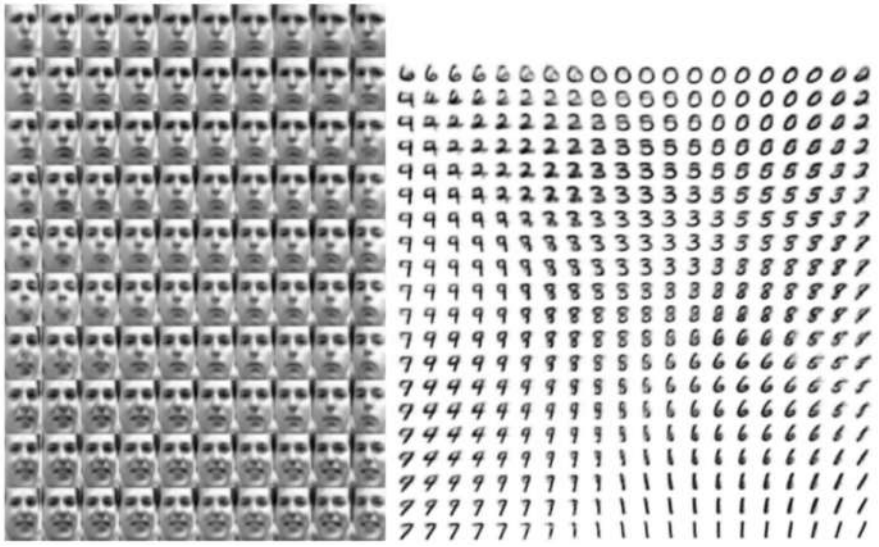


FIGURE 1.6: Examples of generated images using VAEs with 2D codes. Figure taken from [5].

VAEs are trained by maximizing the variational lower bound

$$\mathcal{L}(q) = \mathbb{E}_{z \sim q(z|x)} \log p_{model}(x|z) - D_{KL}(q(z|x)||p_{model}(z)). \tag{1.1}$$

The first summand is a reconstruction log-likelihood loss, and the second is Kullback-Leibler divergence [32]. KL divergence pushes the model prior on latent variables $p_{model}(z)$ and posterior distribution of latent variables $q(z|x)$ close to each other. The idea of VAEs is to train the encoder, which outputs the parameters of distribution $q(z|x)$.

Although VAEs are capable of producing fairly photorealistic results based on learning natural images manifolds (Figure 1.6), their drawback is that samples tends to be blurry [33]. The possible reason behind that is training a VAE with maximum likelihood loss $D_{KL}(p_{data}||p_{model})$ is similar to training a traditional autoencoder with mean squared error, in the sense that it tends to ignore high frequency features (that occupy few pixels).

1.1.1.2 Generative Adversarial Networks

Another type of models used for manifold learning and image generation are Generative Adversarial Networks. Discriminative models learn the conditional distribution $P(y|x)$ in direct manner, i.e. they learn a function which maps the input data x to some desired output label y. Generative models, by contrast, learn the joint probability of input data and class label $P(x, y)$. Generative models can learn the input data structure even in an unsupervised scenario. The idea of GANs was first presented in [24] as a minimax

game between two neural networks:

$$\min_G \max_{D \in \mathcal{D}} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))], \tag{1.2}$$

where $p_{data}(x)$ is true data distribution and $p_z(z)$ is some simple distribution (e.g. $\mathcal{N}(0, 1)$). The first network called "generator" generates samples from the noise. The second called "discriminator" learns to distinguish between samples from the generator and from the training set. The two summands in (1.2) are a standard cross-entropy loss, corresponding to training a binary classifier with a sigmoid output. The classifier is trained to distinguish examples coming from the generator and from the real dataset. Two mini-batches of data are used: the label is 1 for all the examples from the dataset and the label is 0 for all the examples from the generator. The discriminator is trained using both terms and the generator using only one.

The error from the discriminator is backpropagated through the generator. Thus during such simultaneous optimization, both networks learn to perform their task better and better until the generator finally learns to produce samples indistinguishable from real data. In other words, the generator learns to model a manifold of natural images.

Several applications for art creation [6] and photo modifications [7, 34] are based on GAN idea.

Interactive generative adversarial networks (iGAN) were developed in [6]. The network outputs a realistic image similar to a rough sketch drawn by the user. The user sketch and the manifold of natural images are typically very dissimilar. The generator learned by GAN is used as a constraint on the output of the users manipulations with an image, ensuring that the results lie on the learned manifold. A random vector $z \in \mathbb{Z}$ is drawn from a multivariate uniform distribution $Unif[-1, 1]^d$ and the learned generator is used as an approximation of an ideal image manifold $\mathbb{M} \approx \tilde{\mathbb{M}} = \{G(z) | z \in \mathbb{Z}\}$. An Euclidean distance between the images in this latent space is defined:

$$L(G(z_1), G(z_2)) \approx ||z_1 - z_2||^2.$$

Thus linear interpolation in this latent space serves as a way of traversing the manifold.

For a given real photo x_0, the work [6] suggests to first project it on the approximation of the image manifold using the perception loss [35] between intermediate deep neural network activations as a measure of closeness between the generated image and the real photo. Either direct optimization is used or a special projection feedforward neural network is trained. The projection yields the latent vector z_0 of the closest image $G(z_0)$. After that, the image is modified while meeting the constraint of staying within the

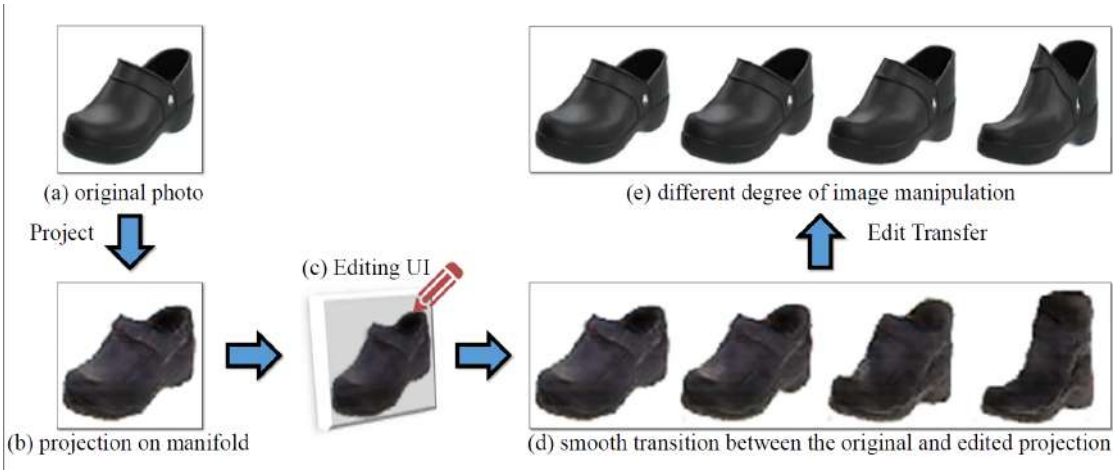


FIGURE 1.7: Image editing on the natural image manifold. The original photo is projected onto a low-dimensional latent vector representation, the color and shape of the generated image are modified, and the same modifications are applied to the original image. Figure taken from [6].

manifold (Figure 1.7) and the image $G(z_1)$ with latent vector z_1 is obtained. Having two latent vectors enables generating the complete manifold curve $[G(z_t)]_{t=0}^{N}$ between $G(z_0)$ and $G(z_1)$ with linearly interpolated latent vectors

$$z_t = \left(1 - \frac{t}{N}\right) z_0 + \frac{t}{N} z_1.$$

The optical flow from $G(z_0)$ to $G(z_1)$ is estimated using that curve and is applied to the initial image x_0, to obtain the final result x_1.

An image-to-image translation concept was suggested in [34]. It covers a lot of different transformations of an image: sketch to photorealistic image, satellite photo to map, and others. In general it is suitable for any pixel-to-pixel transformation. It makes use of conditional GANs [36], with the objective having the form

$$\min_G \max_D \mathbb{E}_{x,y\sim p_data(x,y)}[\log D(x,y)] + \mathbb{E}_{x\sim p_{data}(x),y\sim p_z(z)}[\log(1 - D(x, G(x,z)))].$$

The difference from traditional GANs is that conditional GANs observe both the random noise vector and the image x and learn the mapping $G : \{x, z\} \to y$.

The problem with using simpler L_2 or L_1 loss instead of complicated GAN structure is that it produces blurry results in image generation tasks. However, it can model low frequencies fairly well. Thus [34, 37] also suggest using Markovian discriminator (PatchGAN), which is a combination of GAN discriminator and a standard L_1 loss. The discriminator is restricted to capture only high frequencies, by penalizing only structure at the scale of patches. This discriminator tries to classify each NN patch in an image as a real or a fake one.

FIGURE 1.8: Photo editing in Neural Photo Editor. Top, left to right: Reconstruction, reconstruction error, original image. Bottom: Modified reconstruction, difference between modified reconstruction and $\hat{X}$, output. Figure taken from [7].

An interface called Neural Photo Editor, where the user is able to manipulate the models latent variables in an interpretable way, was created in [7]. The user paints rough modifications to a photo and the network turns these rough paints into a photorealistic image matching the users desires (Figure 1.8). The user paints on the output image, and instead of changing individual pixels, the interface back-propagates the difference between the local image patch and the requested color and takes a gradient descent step in the latent space to minimize that difference. Thus both the image $\hat{X}$ reconstructed from the latent code and the code itself are changing simultaneously. The target input image X is changed and the output Y is produced using a special mask M provides a smoothed and truncated version of the difference between input and reconstruction. The output image is a masked average of the reconstruction, the requested pixel-wise change Δ and the reconstruction error:

$$Y = \hat{X} + M\Delta + (1 - M)(X - \hat{X}).$$

In its Neural Photo Editor [7] uses a model applying both VAE and GAN approaches: Introspective Adversarial Networks. In this approach, the encoder and discriminator are combined into a single network. Four losses are used: ℓ_1 reconstruction loss, VAEs KL divergence in the latent space (1.1), ternary adversarial loss and feature-wise perceptual loss. Ternary adversarial loss, in contrast to standard GAN approach, classifies between the real, generated or reconstructed image, instead of binary real vs. generated classification. Feature-wise perceptual loss is the ℓ_2 difference between the original image and reconstruction in the space of the discriminators hidden layer.

The same perceptual loss was suggested earlier in [29]. In particular, the distance between two images $x, y \in \mathbb{R}^{W \times H \times C}$in their model is

$$\mathcal{L}(x, y) = ||C(x) - C(y)||_2^2,$$

where $C : \mathbb{R}^{W \times H \times C} \to \mathbb{R}^F$ is a differentiable comparator, typically a neural network. The best variety of comparator in their experiments was an AlexNet [38] network trained on the image classification task.

The work [39] concentrates on editing face images, and, apart from adversarial loss, uses additional guidance for a network: priors on a number of intrinsic face properties, such as a morphable model-driven prior on the geometry, a Retinex-based (see [40]) prior on the albedo and an assumption of low-frequency spherical harmonics-based lighting model [41, 42]. The face is separated from the background to traverse the face manifold in the latent space and not to influence the background.

However, the attempts to apply conventional manifold learning with generative architectures to gaze redirection show noticeable sharpness degradation (Figure 1.9). Autoencoders with a combination of ℓ_2 and GAN [24] losses are trained on the Skoltech gaze correction dataset (Section 2.3). The best model has a 200-dimensional latent space and consists of several convolutional and fully-connected layers in the encoder and the decoder. The results do not provide enough perceptual quality and do not meet the needs of gaze manipulation applications. More examples comparing warping to direct re-synthesis are illustrated in Figure 2.2 and Figure 2.3.

1.1.2 Related work on image re-synthesis via warping

To overcome blurriness occurring in traditional generative models, this work focuses on the model utilizing the warping approach (Section 2.1). The idea to do image re-synthesis via warping goes back at least as far as [44]. However, it is only recently that some literary publications suggested obtaining a warping through machine learning. As the work on this thesis was underway, similar ideas started to be discussed in literature. Thus the models suggested here fall within the scope of these discussions. Several papers present a related approach with a deep network predicting warping fields (flow) for image editing.

The authors of [8] exploit the same idea based on the warping field (Section 2.1). They applied idea, similar to the one presented in this work (Chapter 4, Chapter 6), to the task of learning novel view synthesis. The inputs to the network are image and viewpoint transformation coordinates and the output is a warping field applied to the input via

FIGURE 1.9: Examples of reconstructions produced by an encoder-decoder architecture that combines reconstruction ℓ_2 and GAN losses (following the approach in [18, 19, 24, 43]) trained on the Skoltech dataset (Section 2.3). In each pair, the left image is the input and the right is the output. Due to a noticeable loss of fine-scale details and regression-to-mean effect, the result is not good enough for most gaze manipulation applications. Similar issues are observed in [18, 19, 24, 43].

a bilinear sampler. This makes the method similar to the one presented in Chapter 4. However, their encoder-decoder architecture is more similar to the one presented in Chapter 6, where the task is different: having two images, generate warping field from one to another. [8] is yet another work that suggests an approach to learning a manifold of different viewpoints of the same object. The authors generalize the idea of novel view synthesis from a single input to several input images (Figure 1.10). The network called a single view CNN with shared weights is applied to all the input images, and a novel view image is generated from each of them. Aside from the novel view image, the network also generates a confidence mask of the same size as the image, which evaluates the pixel-wise prediction accuracy using this particular input view. The final image is generated as a weighted average of the single novel view, with weights coming from normalizing all masks to sum to one at each pixel location.

[45] suggests synthesizing the flow fields that manipulate the facial expression. The authors use VAE-based architecture with regularization on the latent space representation. Apart from the flow field, they predict a confidence mask for its flow predictions in order to combine multiple source images to generate one output image. Their loss consists of three parts: reconstruction loss, smoothness in the latent space and coherence in the flow space. The flow coherence loss penalizes large flow deviations for close pixels, unless

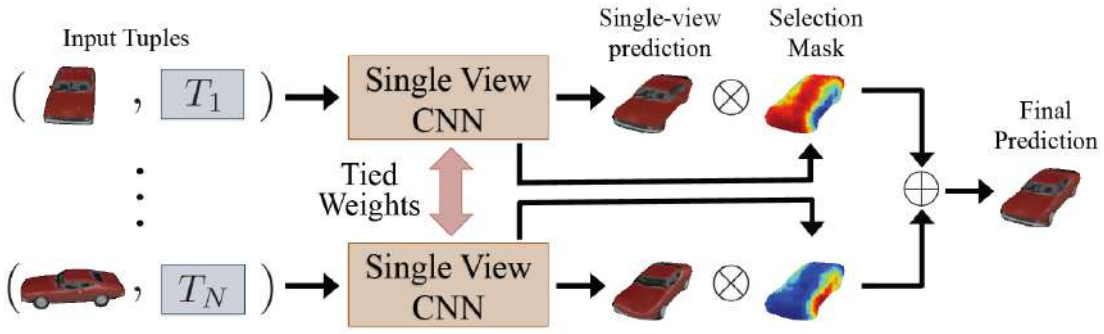


FIGURE 1.10: Multi-view network architecture ($\otimes$: per-pixel product, $\oplus$: per-pixel normalized sum). Shared-weights CNN generates a novel view and confidence mask for each input image. The output is the average of the novel views weighted with masks' values. Figure taken from [8].

there are large deviations in pixel intensities:

$$L_{flow} = \sum_{\mathbf{i}} \sum_{\mathbf{j}} ||\mathbf{F_i} - \mathbf{F_j}||_2^2 \exp(-(\alpha||\mathbf{T_i} - \mathbf{T_j}||_2^2 + ||\mathbf{i} - \mathbf{j}||_2^2)),$$

where $\mathbf{F_i}, \mathbf{T_i}$ are flow and intensity at pixel i. They perform expression manipulation using analogies and interpolation in the latent space. In particular, they calculate the mean change in the latent space on the training database and apply the same change on the latent representation of the query image. They also make the same observation that was found in this work, that upsampling in the flow domain gives sharper results than upsampling in the pixel domain.

The warping flow idea is used for predicting the missed frame in a video sequence in [2]. The training data are triplets of consecutive video frames, two being inputs and the third being a ground-truth which can be in between the frame or the next frame. The warping field and the mask are predicted by selecting between two input frames. The training objective is the combination of the reconstruction loss and total variation regularizers on the flow field and the mask.

The work [2] also puts forward a multi-scale approach to flow prediction. The series of autoencoders works on different scales and produces a series of warping fields, capturing flow on different scales from coarse to fine. The final warping field is obtained by uspampling, concatenation and further convolution of the multi-scale flow fields.

In [46], the warping flow is used to fill in disoccluded pixels, based on the prediction of the visibility map. The authors consider the problem of a novel view synthesis. Firstly, their Disocclusion-aware Appearance Flow Network predicts a warping field as well as a visibility map, which encodes the parts that need to be removed due to occlusion. After

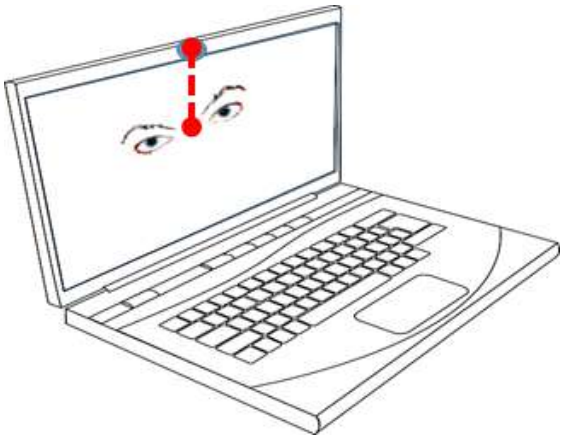


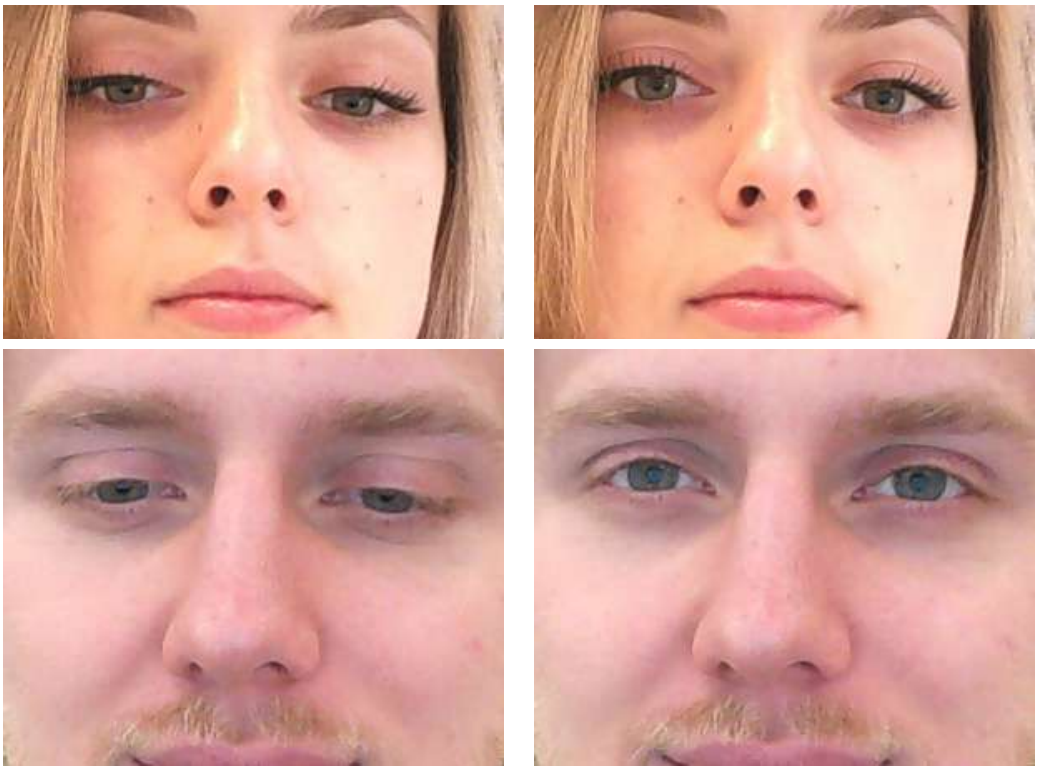
FIGURE 1.11: Eye-to-eye contact cannot be maintained during a videoconference due to a vertical gap between the camera and conference window on the screen. As a result, the person on the screen seems to be looking down while in fact they are looking at your face on their screen.

that, the completion network hallucinates disoccluded regions. They use a combination of reconstruction, adversarial and perceptual loss as an objective.

1.2 Gaze redirection

In this work I concentrated the evaluation of suggested techniques to image synthesis on the problem of gaze redirection. Gaze correction in videoconferencing appears as an essential gaze redirection issue that has been the particular concern of researchers and engineers for a long time. The problem manifests itself as the inability of the people engaged in videoconferencing (the proverbial Alice and Bob) to maintain eye contact, the lack of which is due to the disparity between Bobs camera and the image of Alices face on Bobs screen (and vice versa) (Figure 1.11). Talking to Alices image on the screen, Bob is looking her in the eye, whereas to Alice, he seems to be looking down, because the camera is located above the screen). *Gaze correction* then refers to the process of altering Bobs video stream in a realistic and real-time way, so that Bobs gaze direction as seen by Alice is changed (e.g. redirected upwards) and the eye contact is established (Figure 1.12).

Apart from videoconferencing, there are other important scenarios, where the appearance of the eyes needs to be digitally altered in a way to change the apparent gaze direction. These include talking head-type videos where a speaker reads the text appearing alongside the camera but should redirect their gaze into the camera. Another example is editing photos (e.g. group photos) and movies (e.g. during post-production), making gaze direction consistent with the ideas of the photographer or the movie director (Figure 1.13).



Input Desired output

FIGURE 1.12: Gaze correction in videoconferencing: transformation examples. Top row – gaze redirection by 10°, bottom row – by 15°. A typical angular mismatch is 15° during a videoconference through a consumer laptop with the camera above the screen and with the user sitting at a comfortable distance from the screen.

The solution to the gaze correction problem in videoconferencing would be useful for a broad user audience, including but not limited to designers, filmmakers, HR specialists and job applicants, psychotherapists, language tutors, and TV presenters. Patents on gaze correction and related solutions are held by such companies as Microsoft, Cisco, Samsung, Siemens, NEC, AT&T, Alcatel and others. There has been a substantial commercial demand for gaze correction solutions for decades.

Psychological research has yielded multiple reports underscoring the importance of eye contact. C. Kleinke wrote in his report [47] on eye contact: "Authors of [48] reported a positive correlation between an interviewee's eye contact with an interviewer and estimates made by observers of the interviewee's intelligence. Women rated male dating partners as silent, passive, and inattentive when they gave low levels of gaze during a role-playing interaction [49]. British college students rated a same-sex peer they met in an experiment as more pleasant and less nervous when the person gazed at them continuously rather than not at all [50]." Few image parts have such a dramatic effect on the perception of an image like regions depicting eyes of a person in this image. Humans (and even non-humans [51]) can infer a lot of information about the owner of the eyes, their intent, mood and the world around from the appearance of the eyes

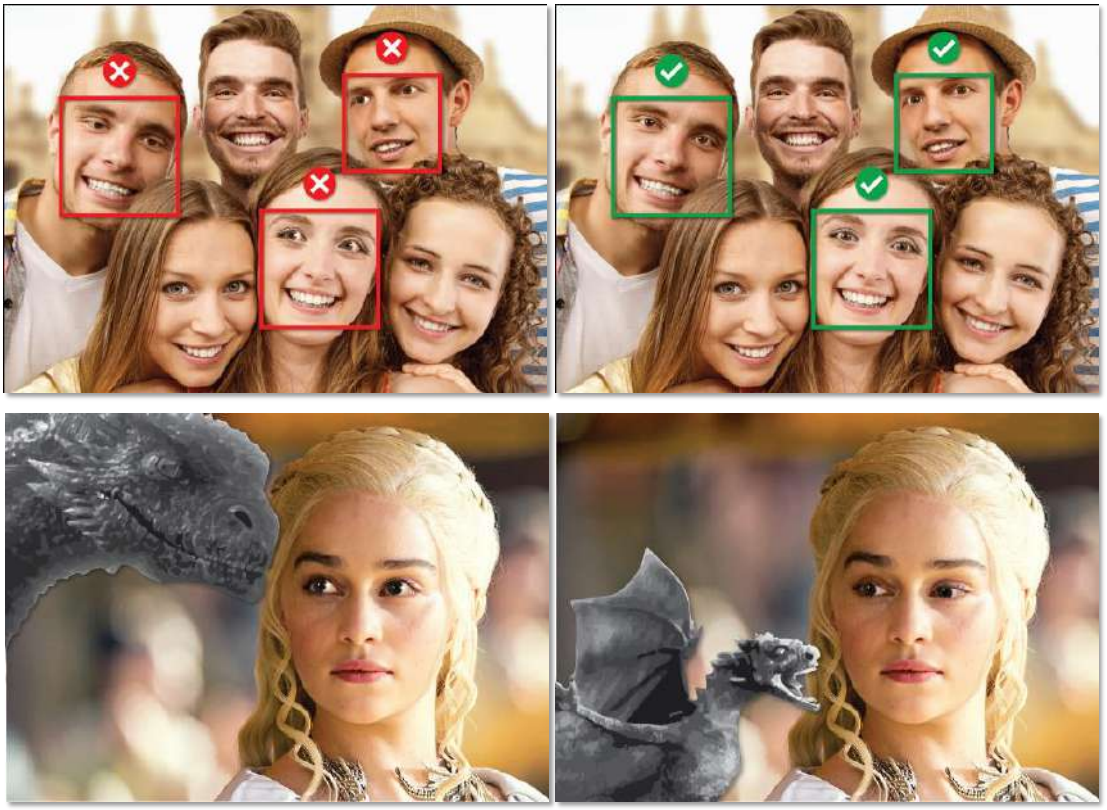


FIGURE 1.13: Gaze redirection for image and video post-processing. Top row: gaze correcting camera for the group of people. Bottom row: post-production of a movie after adding visual effects. CGI character is changed and the actor's gaze is adjusted accordingly. The method is based on constructing a complicated physical model of the eye. Figure taken from [9].

and, in particular, from the direction of the gaze. Overall, the role of gaze in human communication has long been known to be very high [47]. Thus meeting the requirement of realism is particularly difficult due to the well-known *uncanny valley* effect [52], i.e. the sense of irritation evoked by realistic renderings of humans with noticeable visual artifacts, which stems from the particular acuteness of the human visual system towards the appearance of other humans and human faces in particular.

The image re-synthesis methods proposed in the past usually tackle the re-synthesis problem in a general form, striving for universality. In this work I take an opposite approach, focusing on gaze manipulation as a very specific image re-synthesis problem and considering some important real-life applications.

So far, the most successful gaze correction solutions have been relying on additional hardware such as semi-transparent mirrors/screens [10, 53], stereo cameras [54, 55], or RGB-D cameras [11, 56]. Because of the extra hardware dependence, such solutions mostly remain within the realm of high-end conferencing systems and/or research prototypes. Despite decades of research, finding a *monocular* solution that would rely on the

laptop/portable device camera as the only image acquisition hardware remains an open question. The challenge is multi-faceted and involves meeting a host of requirements:

- realism: in order to be applicable in practice, the results should not be perceived by humans as synthetic images;
- having gaze direction/viewpoint position altered sufficiently for re-establishing the gaze contact: Figure 1.12 gives an idea of how much the gaze should be altered, however $2-3^\circ$ of error out of 15° are generally indiscernible by human users;
- real-time performance: several systems suggested in the literature require additional hardware such as a GPU to run in real time; to cater to a broad audience, the system should be capable of operating in real time on a CPU-based consumer laptop and ideally, on portable devices too, and should not require a lot of memory and computational resources.

1.2.1 Previous work on gaze redirection

Fixing the gaze problem in videoconferencing (gaze correction) is the most popular use case of gaze redirection. A number of systems solve the gaze problem using a hardware-driven approach that relies on semi-transparent mirrors/screens [10, 53]. The most popular software-driven approach to gaze correction in videoconferencing amounts to synthesizing 3D rotated views of either the entire scene [54, 55] or the face (which is subsequently blended into the unrotated head) [11, 17]. A two-step mixed software/hardware solution is the most common technique in this category. First, a dense depth map is estimated either through stereo matching [54, 55] or using RGB-D cameras [11, 56]. Then a new synthetic view corresponding to a virtual camera located behind the screen is created in real time. Filling disoccluded regions is a common difficulty encountered in novel view synthesis. As discussed above, reliance on additional hardware represents an obvious obstacle to the adaptation of these techniques.

Although a certain number of purely software-based monocular gaze correction approaches have been suggested [57, 58], most of them have been generally unable to synthesize realistic images and to alter the perceived gaze direction to the extent required. The exceptions are the systems [12, 59] that first pre-record a sequence of frames with a person looking into the camera, and then, at the conference time, replace the eye region with that taken from one of the pre-recorded frames.

Among recent papers, while this study was under way, [13] suggests a similar approach to replace closed eyes with open eyes for photo editing. The work [9] uses a 3D eye model to generate realistic eye images with adjusted gaze.

Paper	Monocular	Does not require GPU	Real-time	Does not require pre-calibration
[11] (2012)	-	-	+	+
[17] (2014)	+	-	+	-
[12] (2010)	+	+	+	-
[55] (2003)	-	+	∼+	+
[54] (2002)	-	+	-	-
[57] (2002)	+	+	-	+
[58] (2003)	+	+	-	-
[56] (2011)	-	+	-	+
[59] (2015)	+	-	+	-
[9] (2017)	+	-	-	+

TABLE 1.1: Comparison of related gaze redirection works' characteristics.

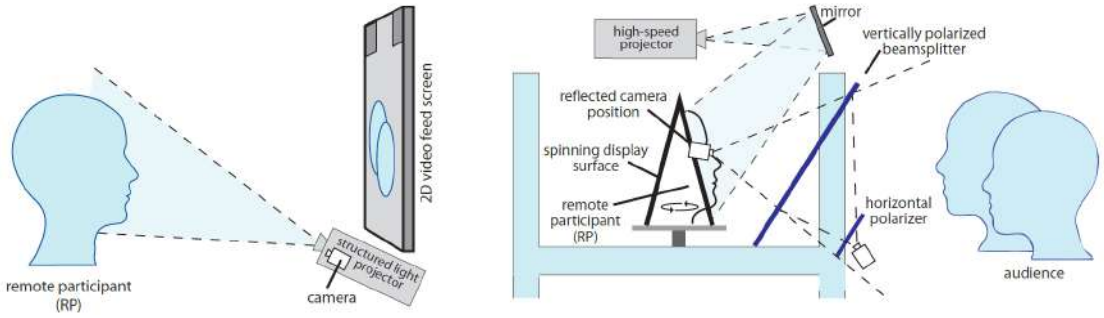


FIGURE 1.14: (Left) a setup for remote participant: the structured light scanning system (120Hz video projector and camera) and large 2D video feed screen. (Right) a setup for audience: the two-sided display surface, high-speed video projector, frontal beam splitter, and 2D video and face tracking camera. Crossed polarizers prevent the video feed camera from seeing past the beam splitter. Figure taken from [10]

.

The table Table 1.1 summarizes some technical aspects of the related software-driven approaches, such as whether a solution is monocular, require GPU, works in real time or require pre-calibration before each session. I do not include in the table the comparison of the performance of the methods because in many cases it is a contentious issue. However, below I describe methods in more detail and give comments on their advantages and drawbacks, including the comments about their performance.

Let us consider some of the systems for gaze correction in more details. Hardware-based approaches are often too complicated and expensive to be used by a wide audience. The system in [10] is designed for. A 3D scanning system scans the remote participant, whose image is shown using a 3D display, and a 2D video system is used for the remote participant to see the audience (Figure 1.14).

1.2.1.1 Gaze correction via 3D view synthesizing

A pair of calibrated stereo cameras located at the top and bottom of the monitor are used in [54]. The method involves pose tracking, view matching and view synthesis. The system initialization requires about 15 minutes of human interaction, which includes horizontally aligning the symmetric facial features such as lips and eyes and using a face modelling tool [60] to acquire a personalized user face model. Pose tracking means tracking a triplet $S = \{\mathbf{p}, \mathbf{q}, \mathbf{m}\}$ for each frame, where $\mathbf{p}$ and $\mathbf{q}$ are points in the camera images and $\mathbf{m}$ is their respective points in the face model. Tracked in its neighborhood are points with salient textures, except for feature points in the non-rigid parts of the face, such as the mouth region. Points $\mathbf{p}, \mathbf{q}$ must satisfy the epipolar constraint [61]:

$$\mathbf{pFq} = 0,$$

where $\mathbf{F}$ is the fundamental matrix, corresponding to the epipolar geometry between the two images. The points are tracked using the KLT tracker [62].

To represent the head pose, a 3×3 rotational matrix $\mathbf{R}$ and a 3D translation vector $\mathbf{t}$ are used. The objective is the sum of re-projection errors of $\mathbf{m}$ to $\mathbf{p}$ and $\mathbf{q}$:

$$e = \sum_i ||\mathbf{p}_i - \phi(\mathbf{A}^0(\mathbf{R}\mathbf{m}_i + \mathbf{t}))||^2 + ||\mathbf{q}_i - \phi(\mathbf{A}^1[\mathbf{R}^{10}(\mathbf{R}\mathbf{m}_i + \mathbf{t}) + \mathbf{t}^{10}])||^2. \tag{1.3}$$

Here $\phi()$ is the pinhole projection, $(R^{10},\ t^{10})$ are the rotation and translation from the second cameras coordinate system to that of the first camera, and $\mathbf{A}^0$ and $\mathbf{A}^1$ are the cameras intrinsic parameters. Reprojection error (1.3) is minimized using the Levenberg-Marquardt algorithm.

A set of good matches in the rigid part of the face is obtained as a result of the 3D head pose tracking. More points and line matching over the entire foreground images are found using several stereo view matching approaches. After that, the novel view is generated based either on view morphing [44] or on hardware-assisted multitexture blending.

The work [55] suggests generating a cyclopean view from two cameras located to the left and right of the screen. The authors develop a stereo matching approach and propose a way to deal with occlusions and hole filling without explicitly constructing a scene 3D model. They generate disparities using a dynamic programming approach, finding the path through a three-plane graph. New labels and cost function are introduced to correctly identify and group occlusions, format the occlusions at the boundaries of foreground objects and ensure inter scanline consistency. Due to different possible camera locations on the screen, they also introduce the way of min-cost surface projection to

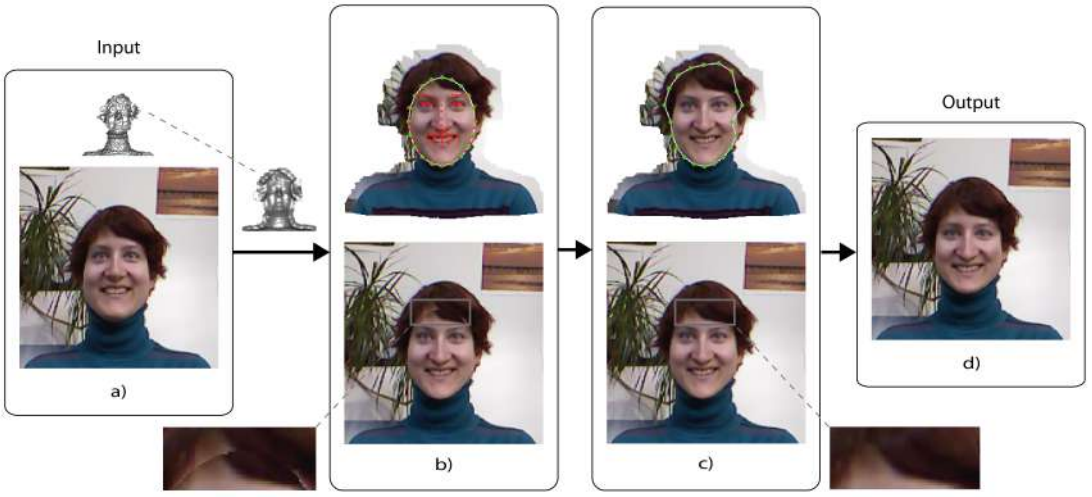


FIGURE 1.15: a) Input color and depth images. b) Top: synthesized image. Bottom: transfer with the use of an ellipse stencil; visible artifacts. c) Seam is optimized; much fewer artifacts. d) Final blending. Figure taken from [11]

.

generate virtual views from arbitrary located cameras directly from the minimum-cost surface obtained during the DP process. Temporal background model construction is applied to decrease temporal artifacts. The gaps on the current frame due to disocclusions can be filled in with values which might be available in previous frames.

The paper [56] suggests a system composed of a time-of-flight depth sensor and a traditional stereo. The depth map from the stereo matching of two images and the depth image from a time-of-flight sensor are joined and refined using the methods from [63, 64]. View morphing [44] in case of a dense correspondence is performed through linear interpolation of projection matrices. A texture splatting technique [65] is used to fill the holes emerging from matching, projection and rounding errors.

The work [11] suggests generat a novel view of the face using an RGB-D image from Kinect (Figure 1.15). Still, some parameters, such as the persons height, need to be manually set before the conference. The user is also advised to refine the result of the 2D placement of the corrected face in the original image (as opposed to initial automatic placement based on coinciding facial features of the two face images, such as eye and mouth positions).

One drawback of this approach is the artifacts resulting from the large size of the face region to repaint and large size of occlusions in this region. The authors suggest applying seam optimization to deal with the issue. A polygonal seam is optimized and made similar in the source and corrected images, so that it appears smoother after blending. The paper mentions another technique named temporal stabilization of discontinuities

in the Kinect geometry. The pitfall of the approach suggested in [11] is the necessity to use Kinect sensor and GPU to compute fast enough for real-time performance.

This approach is further developed in [17] where the novel view is generated using a single web camera with no depth image taken. Instead the method uses monocular real-time approximate fitting of the head model fitted using the facial features extracted from the input image. Before the videoconference, the user should manually relate the feature points to the 3D mesh vertices. During the videoconference, the head mesh is rebuilt based on the facial landmark tracker output [66]. Conversion between the 2D landmark and 3D world coordinate systems is performed using the Laplacian deformation technique [67]. The method uses a similar but slightly enhanced seam optimization technique. Before the session begins, the user is asked to look straight in the camera to record the static texture of the face and correct gaze direction. The texture is then used to fill the occluded vertices of the seam. However, the method proposed in [17] has some limitations, due to multi-perspective nature of the output images, and the availability of a GPU for real-time operation. Pre-recording heavily occluded head parts (under the chin) before each video conference creates further limitations for practical usage of the system.

Monocular setting is also considered in [57] which uses a reference pair of images with target transformation and epipolar geometry for rigid body motion to project the transformation on the input image. Work [58] suggests applying transformation to the geometry without knowing 3D coordinates, but with two different kinds of rotation and eyelids correction post-processing. However, the results presented in these papers fall short of the goal in terms of realism of the generated faces.

1.2.1.2 Gaze correction via eye replacement

The general idea of [12] is to find an accurate position of the eye and to replace it with a pre-recorded image of the eye of the same person with a proper gaze direction (e.g. gaze in the camera). Initial eye detection is performed using an offline database of images with eyes annotated manually according to the model shown in Figure 1.16, which was first suggested in [68].

Once the images with a gaze pointed at the camera have been pre-recorded, eye parameters are found independently for each image. This is done by first localizing the eye corners using the method from [69] and then extracting the SIFT descriptor from the localized region of interest. After that, eye parameters are approximated using the model trained on an offline database (Nearest Neighbor and Support Vector Regression models were used).

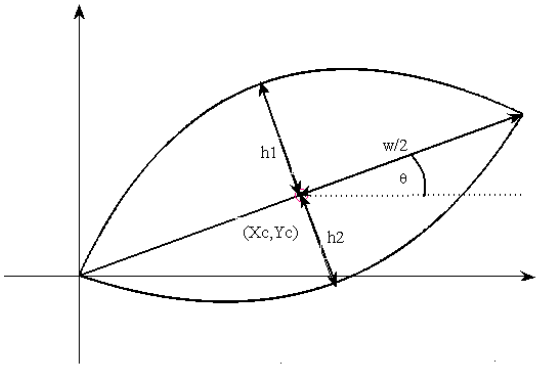


FIGURE 1.16: Eye model consisting of 6 parameters. Figure taken from [12]

.

Next, the 6-parameter space is searched locally to refine the model. For each new parameter value, the closest eye in the offline database is translated, rotated and stretched in accordance with the difference in the parameters. At test-time, the closest matches to straight-looking eyes are found using the cross-correlation measure. Local search in the space of 6 parameters is also used to adjust the eye model for consecutive frames and for blending the eyes found.

The downside of [12] is that while the obtained images achieve sufficient realism, the gaze in the synthesized image remains "locked", staring unnaturally into the camera irrespective of the actual movement of the eyes. Although initially the problem boils down to lack of eye contact, we do not want eye-to-eye contact to be permanent, for fixed gaze would be unnatural in live communication. The only goal is to make up for the vertical gap between the camera and the videoconference window on the screen. A related drawback is that the system needs to pre-record a sufficient number of diverse images of the persons eyes. The systems suggested in this work have none of these limitations.

Another method suggested in [59] is based on replacing the eyes with those looking directly into the camera. Similarly to other techniques, it involves pre-recording a sequence of frames with a person looking directly into the camera and blending the eyes with those from one of the previously taken frames. Thus the method shares the drawbacks of [12]: the gaze is unnaturally directed at the camera all the time and each videoconferencing session requires a set of additional preparatory operations. Face alignment [70] is used to localize eye regions and warp these regions from the destination image to the target one. In order to correct the gaze vertically, the corresponding direction is enlarged by some ratio. Afterwards, post-processing steps are applied, such as reillumination and Laplacian pyramid blending.

1.2.2 Recent work on gaze redirection

Published more recently, [13] suggests an approach similar to that addressed in [12] and replacing closed eyes with open eyes for photo editing (Figure 1.17). The idea is to replace the eyes in the target image with the eyes from the reference image. The authors start with fitting face models to both images, fitting a 3D face model using Principle Component Analysis on a 3D face shape dataset and tracking 2D facial landmarks [66]. Assuming a weak perspective projection, they optimize for a projection matrix and weights in the PCA decomposition using the fitting algorithm [71]. The reconstructed projection is also used to warp a reference image in such a way that the eyes are roughly aligned with the eyes in the target. Then they perform image rectification in terms of lighting, local contrast and overall skin tones using multi-dimensional histogram matching and finally robustly blend the eye regions into the target image.

Another paper, [9] was published after the experimental part of this study had already been complete. Work [9] models the eye region in 3D, recovering the shape and pose of the eyes [72], updates the model parameters in iterative manner, generates a corresponding synthetic eye region image and calculates new updates for the parameters, comparing the synthetic eye region to the observed one. The comparing objective consists of image and landmark similarity terms, as well as of terms corresponding to the eye model (penalizing unlikely eye pose and shape).

The gaze redirection step relies on the fitted eye model. The model parameters are modified to represent the redirected gaze and the optical flow is calculated from these two sets of parameters. The optical flow is used to warp the eyelids. The eyeballs are rendered directly from the new set of parameters. Application to image and video post-processing is illustrated in Figure 1.13.

The objective optimization step in [9] is a complicated high-dimensional non-convex problem, thus a GPU is needed to perform Gauss-Newton iterations. Still, the complete algorithm does not have real-time capability because of the cumbersome model-fitting step.

1.3 Contributions and scope

Chapter 2 describes a general approach to image re-synthesis. The warping field concept with the output image pixels sampled from the input image is introduced. The warping field predictor is learned from the dataset of image pairs describing the target transformation. As regards gaze redirection, I suggest repainting only the vicinity of the eyes,

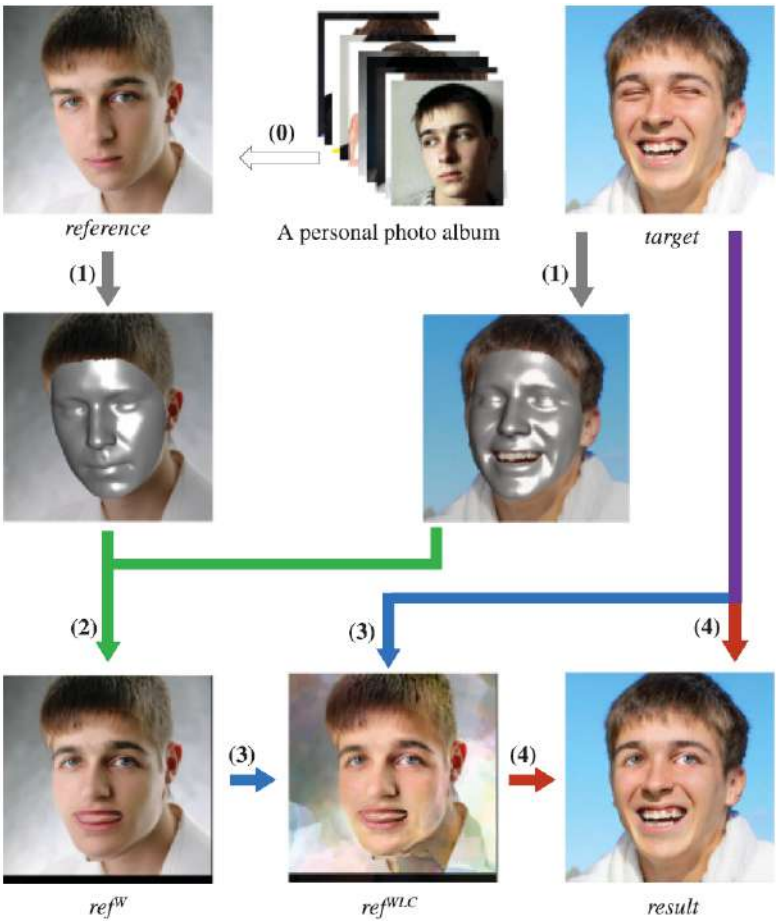


FIGURE 1.17: Eye-editing approach [13]. Steps applied to transfer the eyes: face fitting, eyes' registration via warping, color adjustment, blending. Figure taken from [13].

emulating only the change of gaze direction and keeping the head pose unchanged. A review of facial alignment methods is also given in Section 2.3.1.

Chapters 3-6 deal with four image re-synthesis systems and their application to gaze redirection. The first system [73] described in Chapter 3 is based on a special kind of randomized decision tree ensembles called warping flow forests that are learned in a weakly-supervised manner. For a gaze redirection task, at training time this system observes pairs of input and output images, where each pair contains the face of the same person with a fixed angular difference in the gaze direction. It then learns to synthesize the second image in a pair from the first one by predicting a warping flow field. After learning, the system acquires the ability to redirect the gaze of a previously unseen person by the same angular difference as in the training set. The system synthesizes realistic views with a gaze systematically redirected by $10-30$ degrees in the experiments. At test time, the system accomplishes gaze correction using simple pixel replacement operations that are localized in the vicinity of the persons eyes, thus achieving high computational efficiency. My implementation runs in real time on a single core of a laptop. Although the results are of high perceptual quality (see Section 5.3.4), the system still leaves some

room for improvement and artifact reduction. Although less critical, another deficiency of the system is a relatively large memory footprint of the learned models.

The second system (Chapter 4, [74]) is based on a deep feed-forward architecture that combines several principles of operation (coarse-to-fine processing, image warping and intensity multiplication). The architecture is trained end-to-end in a supervised way using a specifically collected dataset that depicts the change of appearance that occurs as the gaze is redirected by different angles. Qualitative and quantitative evaluation demonstrates that the suggested deep architecture can synthesize very high-quality eye images, as required by the nature of the applications, and does so at several frames per second. The quality of the results is higher than that in the first system, but the efficiency falls short of real-time CPU operation, making the method impractical for video-conferencing with most commonly used consumer devices. Such approach is still useful for image and video-editing application scenarios outlined above. The system also contributes to an actively-developing field of image generation with deep models.

The third system (Chapter 5, [75]) combines the advantages of the previous two. Similarly to the first system, it is based on a randomized decision forest which in this case is trained in a traditional fully-supervised manner. To obtain such supervision, I use an output of the second system to effectively have the deep architecture teach the random forest. At training time, the new system observes images and the corresponding warping flow that is estimated by the deep model and learns to produce the flow for previously unseen images using a regression random forest. At test time, the system outputs a result by applying the flow predicted by the random forest to the input image. As shown by the experiments described in Section 5.3, the trained forest attains nearly the same quality as the teacher network, outperforming a weakly-supervised random forest (from the first system). At the same time, this system runs in real time on a single core of a CPU. Moreover, the resulting models have a much smaller memory footprint as compared to the weakly-supervised forest.

All the three systems are trained in a supervised case, based on a large dataset of images labeled with a gaze direction angle. Acquiring of such a database is comparatively expensive, as it requires significant time per user, adherence of proper instructions, and still contains some noise in labels (Section 2.3). The fourth system (Chapter 6) is dedicated to unsupervised and semi-supervised scenarios. In this case the first model learning step is unsupervised. A large dataset of *unlabeled* images is utilized, which are sampled from sequences of eye images of different people with varying and unknown gaze directions. Such data is much easier to get (Section 2.3). The goal of unsupervised learning is to train a decoder network which maps input images to a shallow representation space, and a decoder network which learns to construct a warping field from the first image in

Method	Inputs	Training data	Computational demands	Quality of results
Weakly-supervised Random Forest (Chapter 3, [73])	Image and landmarks, image is processed at native resolution, gaze is changed in fixed direction	Pairs (input + ground truth) of images with fixed gaze direction difference, landmarks for input image	Real-time performance on a consumer laptop ($\approx$ 40 FPS), $\approx$ 100 Mb RAM on storing a model	Beats the baseline by a large margin, worse than other methods, but performance is still good in User study
Deep Warp (Chapter 4, [74])	Image, landmarks and redirection angle, image is processed on fixed training spatial scale	Pairs (input + ground-truth) of images and gaze direction difference between them (angles along x and y directions), landmarks for input image	Performance up to 20 FPS on a GPU	The best results in quantitative comparison and User Study
Random Forest supervised by Neural Network (Chapter 5), [75]	Image and landmarks, image is processed at native resolution, gaze is changed in fixed direction	Input images with landmarks, flow fields corresponding to redirection of input images by some fixed angle	Real-time performance on a consumer laptop ($\approx$ 40 FPS), $2-3$ Mb RAM on storing a model	Very close to the results of a teacher Deep Warp model in both User study and quantitative comparison
Semi-supervised Neural Network (Chapter 6)	Image, landmarks and redirection angle, image is processed on fixed training spatial scale	Pairs of images without known difference in gaze direction at training time; input image and analogy pairs from labeled part of dataset with target difference in gaze direction at runtime	Performance up to 20 FPS on a GPU	Outperforms Deep Warp approach given large unlabeled dataset with small labeled part

TABLE 1.2: Comparative analysis of gaze redirection systems suggested in this work.

the pair to the second based on their representations. Having the encoder and decoder learned, the system can redirect the gaze of a previously unseen eye in a semi-supervised way, based on a relatively small collection of eyes with gaze direction labels. The labeled part of the dataset is used to pick up analogy images. The latent representation of a test eye is modified additively based on the representation of analogies. After this, the decoder network will be able to estimate a warping field from the test eye to an unknown target eye with a redirected gaze.

Table 1.2 compares the four methods in terms of inputs, training data, speed and quality. Each Chapter gives a review of the related work: Random Forests and their application in Computer Vision in Section 3.1, Neural Networks in Section 4.1, teacher-student architectures in Section 5.1 and analogy models in Section 6.1. Chapter 7 offers some final comments about the key aspects of the work presented.

1.4 Work overview

1.4.1 Contributions and scientific novelty

The key contributions of the work are the approach to image re-synthesis based on the warping field and four methods for solving this problem in different scenarios. In particular, the authors personal contributions are detailed below:

- The approach to image re-synthesis is formulated as the task of learning the warping field predictor from the dataset of examples. The warping model is applicable to problems without large dis-occlusions or changing colors from input to the output so that warping model is capable to model the target transformation.
- Several methods of learning a warping field predictor are suggested:
  - weakly-supervised random forest;
  - fully supervised random forest;
  - semi-supervised neural network for embedding learning.
- Quantitative and qualitative comparisons of the methods are performed on the gaze redirection task, including the user study.
- A method for collecting and preprocessing the dataset for gaze redirection is suggested. The dataset consists of pairs of images describing the gaze redirection and used for the learning of suggested methods.

The novelty of the general approach to image re-synthesis lies in the learning of the warping field of displacement vectors for gaze redirection from the dataset of images. The idea to do image re-synthesis via warping goes back as far as [44]. However, the warping field in [44] and similar works is constructed from two basis views of a static scene using geometrical projection properties. No references to the use of machine learning for obtaining the warping field have been found in the literature published prior to this work which suggests new methods for learning a warping field predictor.

Warping field prediction as such pertains to learning tasks with a structured output. Section 3.1.3 describes a series of works that apply random forests to predict the structured output in a computer vision application. The work [76] predicts the optimal filters in the tree leaves. The novelty of the weakly-supervised forest suggested in Chapter 3 is that error distribution over all warping vectors and not only a single optimal warping vector can be stored in the leaves. Another novelty is that ensembling of several trees in the forest is based on summing up these distributions.

The novelty of the DeepWarp approach is that deep architecture is learned from a dataset of input-output image pairs to produce a warping field. By the time the work was published, only direct regression of the target image pixels had been reported in the literature (see Section 1.1.1).

The teacher-student approach was used to train a faster neural network from large and slow models: a larger neural network [77], or an ensemble of networks [78, 79]. The architecture suggested in Chapter 5 offers a novel teacher-student combination, with the weakly-supervised neural network acting as the teacher and the random forest as the student. The neural networks representation power helps to achieve high accuracy, while the random forest is an architecture capable of fast operation at test time, which makes it suitable for real-time implementation on a CPU

The work [18] suggests a deep-analogy-making model. However, the gaze redirection model suggested in Chapter 6 uses deep embedding learning and does not require the knowledge of transformations in the form of analogy-forming quadruplets. The novel nature of the method is that it uses a combination of unsupervised training on pairs of images without gaze labels and image analogy making in the latent space as applied to realistic image re-synthesis.

1.4.2 Academic and non-academic impact

Practical value. The suggested methods are useful for the image synthesis tasks where the target transformation is described by a training set of input-output pairs of images, for example, changing facial features or video interpolation. As applied to gaze redirection, the forest-based methods are suitable for real-time monocular gaze correction in videoconferencing, where eye-to-eye communication is impossible because of the vertical gap between the camera and the eyes of the person on the screen. Other applications include post-processing of images and video for photo-editing and movie post-production, teleprompting and TV presenting, where gaze redirection can be used to ease the process of using text notes.

Approbation, publications and personal contribution. The results of this work were presented at multiple conferences and scientific seminars:

1. Computer Vision and Pattern Recognition Conference, 2015, Boston, poster and demo;

2. European Conference on Computer Vision, 2016, Amsterdam, poster and demo;

3. Seminar of the Laboratory 11, Institute for Information Transmission Problems RAS, 2016, Moscow;
4. Seminar named after M.R. Shura-Bura, Keldysh Institute of Applied Mathematics RAS, 2017, Moscow;
5. Computer Vision seminars, Yandex School of Data Analysis, 2014, 2015, 2016, Moscow;
6. SkoltechOn conference, Skolkovo Institute of Science and Technology, 2015, Moscow;
7. Machines Can See, Computer Vision conference, 2017, Moscow, demo;
8. Open Innovations Forum 2015, Moscow, demo;
9. Skolkovo.ai conference, 2016, Moscow, demo.

The results were published in the following papers:

1. Kononenko, D., Lempitsky, V. (2015). Learning to look up: Realtime monocular gaze correction using machine learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4667-4675).

   The idea of gaze redirection by re-synthesis of eye vicinity using the warping field and the setting for the dataset collection, described in Chapter 2, are suggested by the author. Also, the author suggested the ideas of learning method, implemented the method and conducted the computational experiments on weakly-supervised forest, described in Chapter 3.

2. Ganin, Y., Kononenko, D., Sungatullina, D., Lempitsky, V. (2016). DeepWarp: Photorealistic image re-synthesis for gaze manipulation. In European Conference on Computer Vision (pp. 311-326). Springer International Publishing.

   The authors personal contributions include an experimental setup and data preparation for the DeepWarp approaches, comparison of the DeepWarp and forest-based approaches, and setup and implementation of the user study described in Chapter 4.

3. Kononenko, D., Ganin, Y., Sungatullina, D., Lempitsky, V. (2017). Photorealistic Monocular Gaze Redirection Using Machine Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (Epub ahead of print).

   The authors personal contributions are the teacher-student architecture, its implementation, computational experiments and user study, described in Chapter 5.

Along with the papers listed above, the authors further contributions include the semi-supervised flow learning concept, implementation and computational experiments described in Chapter 6.

The following patent was obtained:

- Daniil Kononenko, Victor Lempitsky, Method for correction of eye images using machine learning. RU Patent 2596062 , August 8, 2016.

  The license for the developed technology was purchased by the RealD company [80].

Chapter 2

Machine-learning-based image re-synthesis

All the suggested systems use some form of supervised learning, as the training proceeds by observing multiple pairs of input and output images. The main challenge of such approach is to devise learning methods that can generalize to instances and imaging conditions unseen during training, and all the systems that I discuss below achieve such generalization by learning to predict the warping field rather than the output image directly. The second system (Chapter 4) corrects the warped output with per-pixel brightness modification (while still trying to achieve the redirection mostly by warping). I will now discuss the elements common to all the systems

2.1 Image re-synthesis by pixel-wise replacement

The target transformation is defined by a dataset of training image pairs

$$\{I_i(x,y,k), O_i(x,y,k)\}_{i=1}^{L},\ x \in 1,\ldots,M, y \in \{1,\ldots,N\}, k \in \{1,2,3\}. \tag{2.1}$$

The method needs to alter pixels of input image to emulate transformation. I rely on machine learning to accomplish this. The approach of direct regression of pixels of output image have problems with synthesizing high-resolution images ([18, 19, 24, 43], Figure 1.9, Figure 2.2, Figure 2.3). Thus, the warping-based approach is suggested in this work. A 2D offset vector $(u(x,y), v(x,y))$ is obtained for each pixel (x,y). The final value of the pixel $O(x,y)$ in the output image O is then computed using the following simple formula:

$$O(x,y) = I\left(x + u(x,y), y + v(x,y)\right)\ . \tag{2.2}$$

In other words, the pixel value at (x, y) is "copy-pasted" from another location determined by the *warping flow* vector (u, v). In case of color RGB image, the operation is performed channel-wise:

$$O(x, y, c) = I\left(x + u(x, y), y + v(x, y), c\right) \; \forall c. \tag{2.3}$$

The operation could be easily generalized to a case when offsets are non-integer numbers. I define a warping field, consisting of two maps, corresponding to the set of 2D offset vectors for the whole image:

$$\mathcal{F}(x, y) = \left(u(x, y), v(x, y)\right). \tag{2.4}$$

The operation, which samples an image I in (possibly non-integer) offset coordinates, resulting in the output image O, is denoted as

$$O = \mathbf{S}(I, \mathcal{F}). \tag{2.5}$$

The values in non-integer coordinates are interpolated, given all pixel intensities. One of the common interpolation methods is a bilinear interpolation [81]. The value of the intensity is interpolated depends on the distance to four closest point in pixel grid Figure 2.1. If we denote $\tilde{x} = x + u(x, y)$, $\tilde{y} = y + v(x, y)$, $x_1 = \lfloor \tilde{x} \rfloor$, $x_2 = \lceil \tilde{x} \rceil$, $y_1 = \tilde{y} \rfloor$, $y_2 = \tilde{y} \rceil$, then the value of interpolated intensity is

$$\begin{aligned} O(x, y) =& I\left(x + u(x, y), y + v(x, y)\right) \approx \\ & (x_2 - \tilde{x})(y_2 - \tilde{y}) I(x_1, y_1) + (\tilde{x} - x_1)(y_2 - \tilde{y}) I(x_2, y_1) + \\ & (x_2 - \tilde{x})(\tilde{y} - y_1) I(x_1, y_2) + (\tilde{x} - x_1)(\tilde{y} - y_1) I(x_2, y_2). \end{aligned} \tag{2.6}$$

A bilinear sampler (a sampler (2.5), using bilinear interpolation (2.6)) is piece-wise differentiable [82]. Thus it could be incorporated into deep architecture with end-to-end training based on gradient descent. It is further discussed in Chapter 4 and Chapter 6.

The methods presented in this work in the following chapters are trained at a fixed spatial scale. At test time, some of them can be applied on the native resolution of the input image (forest-based methods), while other require the same fixed spatial scale, as during training (neural network-based methods) – see table 1.2 for comparison. Thus, for neural network-based methods the input is rescaled to match the resolution at training time. The standard approach after the pass through the neural network would be to upsample the result. However, having the warping field, instead of upsampling the result itself, I upsample the resulting warping flow using bilinear interpolation (2.6), and apply

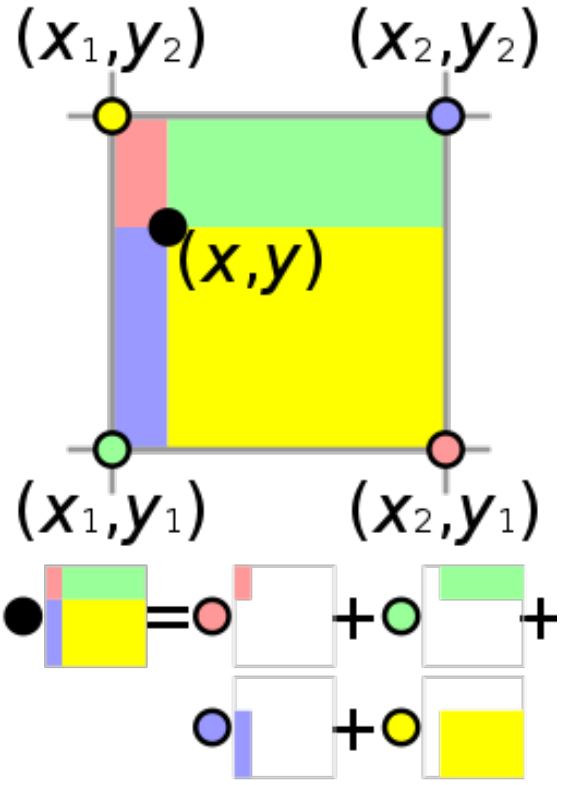


FIGURE 2.1: Visualization of bilinear interpolation. The value at the black spot is the sum of the value at each coloured spot multiplied by the area of the rectangle of the same colour, divided by the total area of all four rectangles. Image taken from [14].

the upsampled flow to the input image in native resolution (2.5). Such approach results in more sharp images.

A warping "copy-pasting" procedure (2.2) is a restriction on the transformation family. It introduces natural regularization in suggested method. This approach ensures that the pixels of the output are copied from the input rather than "invented". I present the example comparison of warping approach versus direct regression on a gaze redirection task in Figure 2.2 and Figure 2.3. While unregularized model suffers from noticeable fine-scale details dropping and regression-to-mean effect, warping approach produces photorealistic results. Also, warping model shows comparable results on training and testing sets, while non-warping model suffers from overfitting. The model and description of training process used in this comparison are fully described in Chapter 6. I refer reader to this chapter for the details, describing the model briefly here.

Both warping and not-warping models are trained on pairs of eyes of the same person with different unknown gaze direction. Both eyes are encoded into low-dimensional space, and the encodings are stacked together. The warping model is then learned to produce the warping field from the first eye to the second from these stack of encodings,

FIGURE 2.2: The comparison of warping and non-warping approaches, training set. Columns from left to right: input image, ground-truth image, the result of warping model, the result of non-warping model. Significant effects of dropping fine details, noise and regression-to-mean are decreased using a warping-based approach (2.2)

.

while non-warping model directly outputs the second eye from its encoding, being thus a conventional autoencoder.

However, in some cases the warping field is not enough to reproduce the ground-truth image because of the lack of necessary pixels in the input image Figure 4.3. These cases and a solution are further discussed in Section 4.4.

The warping field approach makes the learning problem weakly-supervised, because we do not have the warping field (2.4) for the training pairs. Thus, the learning method should be developed to handle such weak supervision. Examples of warping fields for gaze redirection are visualized in the Figure 5.1.

FIGURE 2.3: The comparison of warping and non-warping approaches, testing set. Columns from left to right: input image, ground-truth image, the result of warping model, the result of non-warping model. In comparison to train results Figure 2.2, not-warping model exhibits significant overfitting.

.

2.1.1 General pipeline for image re-synthesis

Based on the warping field approach, the following pipeline for image re-synthesis problem is suggested:

1. The target transformation is described by a set of training pairs $\{I_i, O_i\}_{i=1}^{L}$ (2.1).

2. Learn a warping field predictor:

$$p : M \times N \times 3 \rightarrow M \times N \times 2.$$

   The methods to learn such a predictor, suggested in Chapter 3, Chapter 4, Chapter 5, Chapter 6, are the main topic of discussion of this work.

3. At test time for a new image predict the warping field $\mathcal{F} = p(I)$.

4. Generate new image via bilinear sampling of input image $O = \mathbf{S}(I, \mathcal{F})$ (2.2).

In the rest of my thesis I concentrated on the task of gaze redirection, however, all suggested methods to learn a predictor are applicable to this general image re-synthesis pipeline. The restriction on the image re-synthesis problem where the warping is applicable, is that pixels of the output image are contained in the input so that the family of warping transformations is rich enough to model the target transformation.

2.2 Gaze redirection by re-synthesis of eyes vicinity

For an input image frame, most previous systems for gaze correction synthesize a novel view of a scene from a virtual viewpoint co-located with the screen [54, 55, 57, 58]. Alternatively, a virtual view restricted to the face region is synthesized and stitched into the original video stream [11, 17]. Novel view synthesis is however a challenging task, even in constrained conditions, due to such effects as (dis)-occlusion and geometry estimation uncertainties Figure 1.15. Stitching real and synthetic views can alleviate some of these problems, but often results in distortions due to the multiperspective nature of the stitched images.

I suggest not to attempt to synthesize a view for a virtual camera. Instead, methods which are described in this work emulate the change in the appearance resulting from a person changing her/his gaze direction by a certain angle (e.g. ten degrees upwards), while keeping the head pose unchanged (Figure 2.4). Emulating such *gaze redirection* is also challenging, as it is associated with

- complex non-rigid motion of eye muscles, eyelids, and eyeballs,
- complex occlusion/dis-occlusion of the eyeballs by the eyelids,

Input Desired output My output

FIGURE 2.4: The setting for monocular gaze redirection. Left – an input frame with the gaze directed below the camera. Middle – a "ground truth" frame with the gaze directed 15 degrees higher than in the input. Given an input image and the desired change in angle and direction ("15 degrees higher") my method aims to produce an image that for human perception is as close to ground truth as possible.The result of one of the proposed systems is shown on the right. The top row is the example of system proposed in Chapter 3 applied to an image from a Columbia Gaze dataset [83]. The bottom row is the example of system from Chapter 5 applied to an image from a Skoltech dataset Section 2.3. In this particular example, the computation time of the method is 8ms for the top row and 5 ms for a bottom row on a single laptop core (excluding feature point localization). Such speed makes both systems suitable for real-time use in videoconferencing.

- change in illumination patterns due to the complex changes in normal orientation.

The key insight is that while the local appearance change associated with gaze redirection is complex, it can still be learned from a reasonable amount of training data. An additional, rather obvious advantage of gaze redirection as compared to view synthesis, is that gaze redirection can be performed locally in the vicinity of each eye and thus affects a small proportion of pixels in the image. At runtime all suggested methods localize each eye and then perform local operations with eye regions pixels to accomplish gaze redirection.

2.3 Database collection and eye localization

The publicly available Columbia Gaze dataset [83] for gaze tracking application includes 56 persons and five different head poses (0°, ±15°, ±30° horizontally). For each subject and each head pose, there are 21 different gaze directions: the combinations of seven horizontal ones (0°, ±5°, ±10°, ±15°) and three vertical ones (0°, ±10°). Taking the

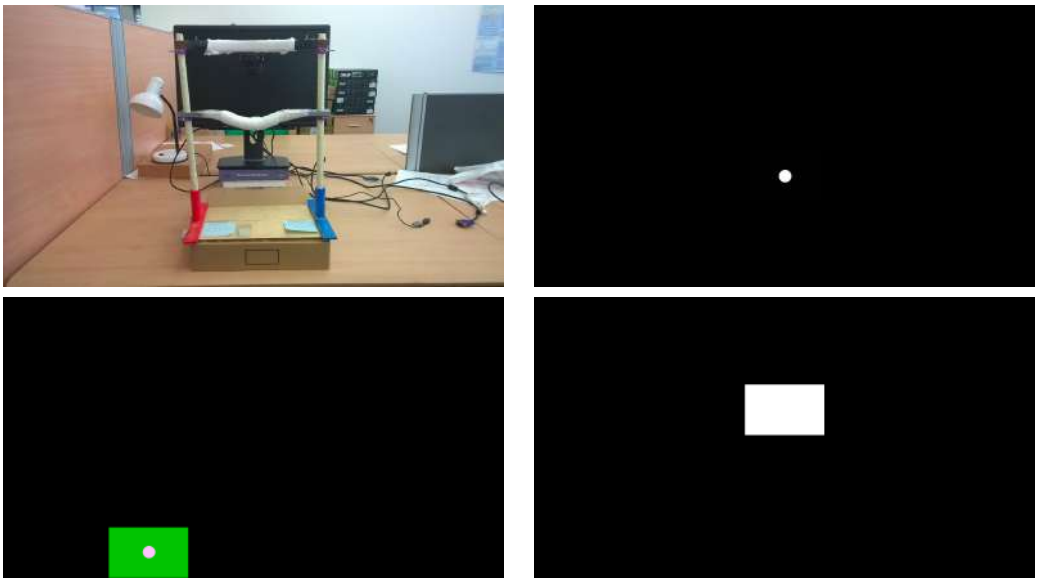


FIGURE 2.5: Dataset collection process. A person is asked to follow the point on the screen with the eyes only, without head movements. To minimize possible head shaking, a special stand is used. (Left top): the setting, (right top): the point is moving, (left bottom): on the borders the point is stationary for a few seconds, so that a person can blink; (right bottom): when the point passes behind the camera, the image of point is enlarged.

pairs with the same parameters except for the vertical gaze direction, I draw training samples for learning to correct gaze by 10 degrees.

However, there are several limitations with Columbia Gaze dataset. The first one is the poor variety in vertical redirection angles. The main testbed for gaze correction in videoconference is redirection on 15° upwards Figure 1.12, and no such examples could be drawn from Columbia dataset. Also, images in this dataset are only loosely registered, which ntroduces an additional error and add a need for additional registration procedures Section 3.2.4. And finally, there are not enough training examples to learn a fully generalizable model.

To avoid these limitations, a Skoltech Dataset of videos of around 150 people was collected. During recording, to minimize head movement, a person places her head on a special stand and follows with her gaze a moving point on the screen in front of the stand. The sequence of frames synchronized with the point position, from which we can deduce the gaze direction, is recorded using a webcam mounted in the middle of the screen Figure 2.5. The examples from the one sequence in the dataset are presented in Figure 2.6.

About 200 frames for one video sequence are recorded. The angular range is 36° in vertical direction and 60° in horizontal direction. I manually exclude bad shots, where a person is blinking, not changing gaze direction monotonically as anticipated or moving

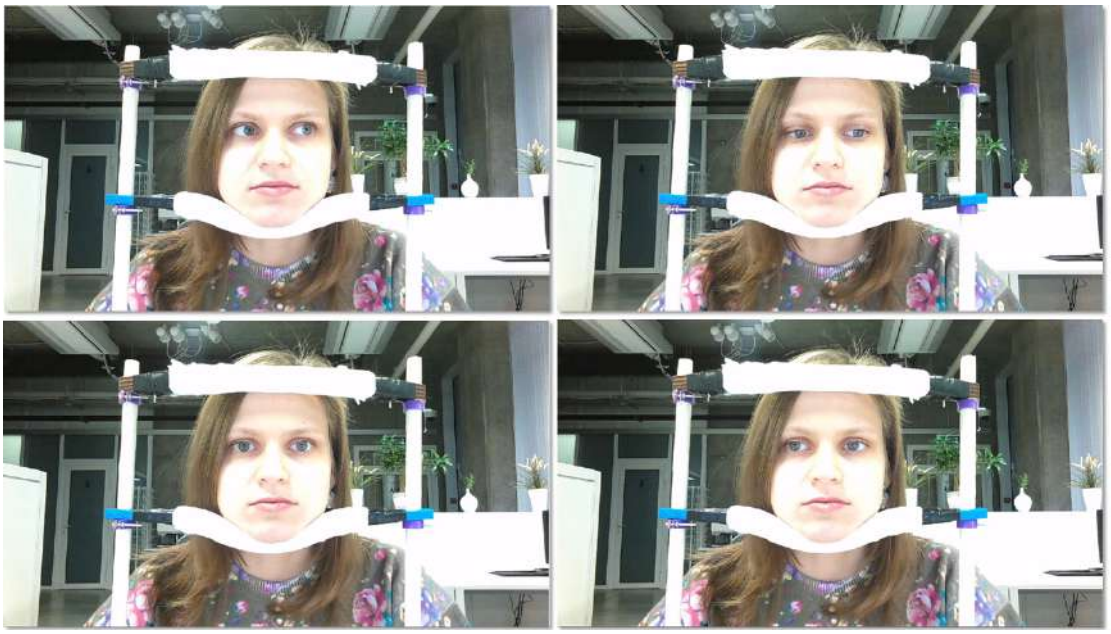


FIGURE 2.6: Examples from the dataset. A data where a person is changing gaze direction only with the eyes, without head movements, is collected.

head. For each person I record $2-10$ sequences (500 sequences total), changing the head pose and lighting conditions between different sequences. From each sequence, one can draw about $50-80$ *training pairs* for a certain angular difference in gaze directions. Each training pair can be regarded as a training example for supervised learning.

All systems suggested in this work start by localizing the eye regions and then achieving redirection by localized processing. Training samples are cropped using the eye localization procedure. The eye localization step within my system is standard, as I use an off-the-shelf real-time face alignment library (e.g. [70, 84, 85]) to localize facial feature points. As the gaze-related appearance change is essentially local to the eye regions, all further operations are performed in the two areas surrounding the two eyes.

For each eye, I thus focus on the landmark points $\mathbf{l}_1 = (l_1^x, l_1^y), \mathbf{l}_2 = (l_2^x, l_2^y) \dots \mathbf{l}_N = (l_N^x, l_N^y)$ corresponding to that eye (in the case of [84] there are $N = 7$ feature points). I compute a tight axis-aligned bounding box $\mathcal{B}'$ of those points. After this, I define the final bounding box $\mathcal{B}$ having the same center as $\mathcal{B}'$ and having the width W and height H that are proportional to some characteristic radius Δ (i.e. $W = \alpha\Delta$, $H = \beta\Delta$ for certain constants α, β). The bounding box $\mathcal{B}$ is thus covariant with the scale and the location of the eye, and has a fixed aspect ratio $\alpha : \beta$. This makes it sufficient to use the same distance to the camera for all dataset images.

I suggest two variants of defining characteristic radius Δ:

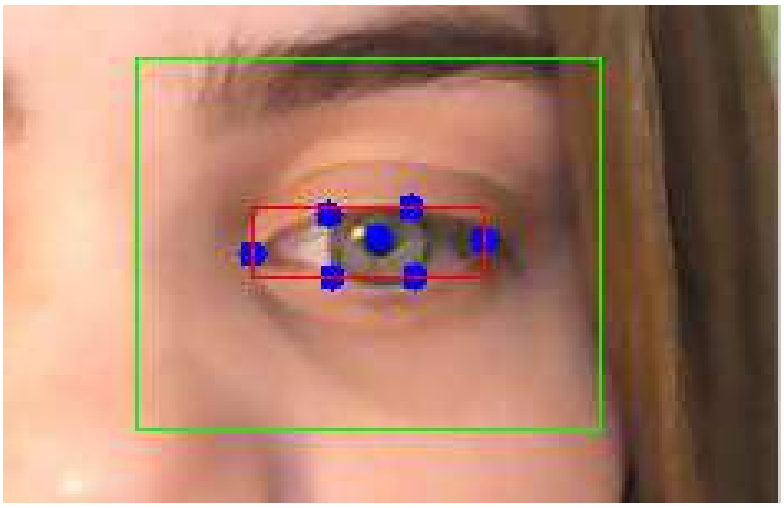


FIGURE 2.7: Illustration of eye localization (variant (2.8)). Blue: eye landmarks, red: tight bounding box around landmarks, green: bounding box, enlarged proportionally to the distance between corner landmarks, to which suggested approach restricts all modifications.

- square root of square of tight bounding box $\mathcal{B}'$:

$$\Delta = \sqrt{\text{Area}(\mathcal{B}')}; \tag{2.7}$$

- the width of the tight bounding box, which equals the distance between the corners of an eye:

$$\Delta = ||l_1^x - l_4^x||. \tag{2.8}$$

An example of the localization process (for the second variant) is shown in Figure 2.7.

In general, the second variant (2.8) results in more uniform eye scaling. In Chapter 3 I provide results for the first variant (2.7), and in the next chapters – for the second variant.

I incorporate left and right eyes into one dataset, mirroring right eyes. At test time, the same predictor could be used for left and right eyes, mirroring the results. Examples of training image pairs are shown in Figure 2.8.

Using the eye tracker could benefit the data collection. It can validate that user actually followed the point on the screen, and, otherwise, the bad shots could be removed from the training data. Eye tracker was not used in this work, therefore the process of removing outliers was more complicated, including manually looking through images with high training error.

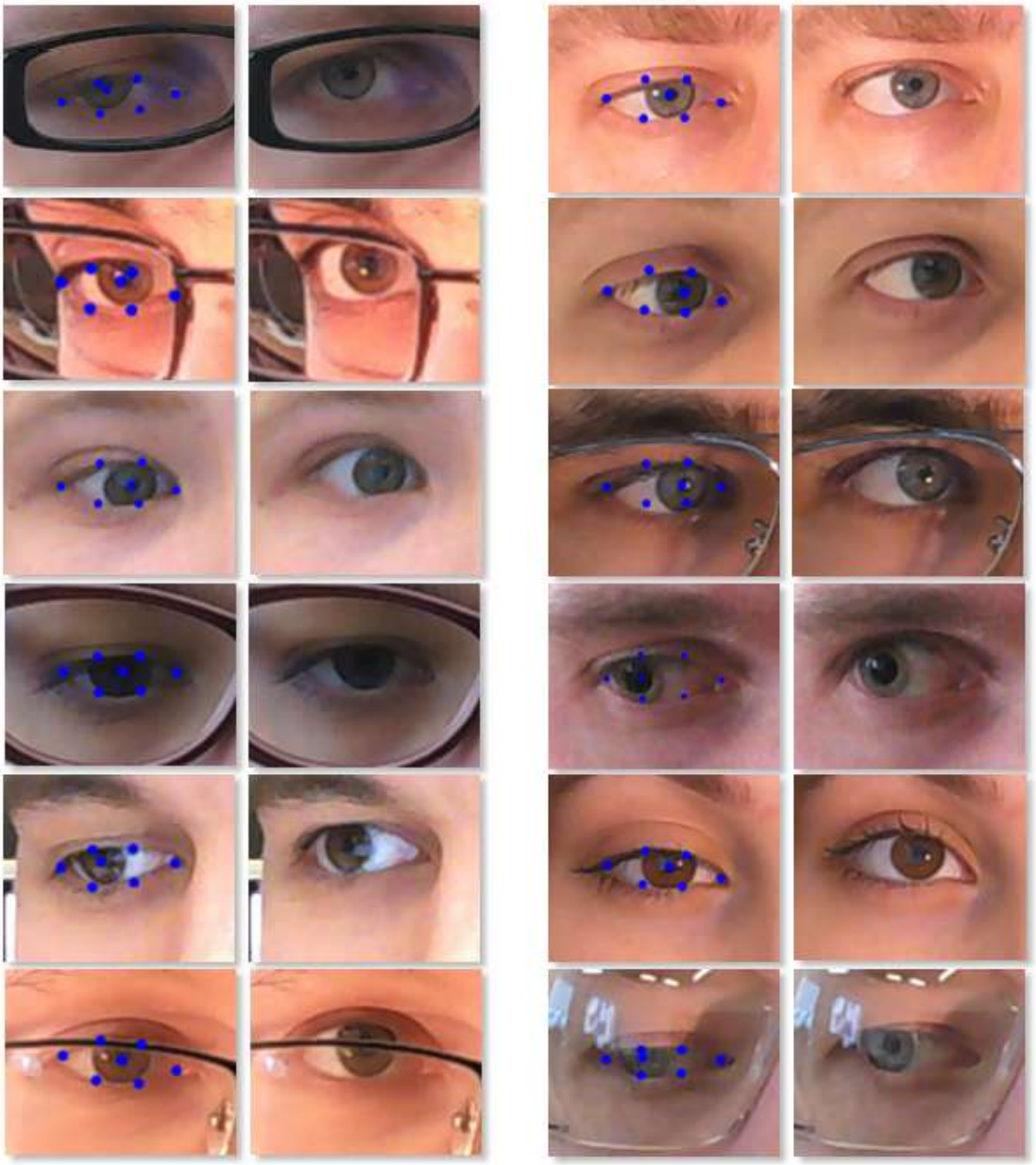


FIGURE 2.8: Examples of training data.

2.3.1 Related work on Facial Alignment

To crop eye images from training examples, I use facial alignment software [70, 84, 85]. I also use tracked landmarks as additional features for a training model. At this section, I give a review of a related work on facial alignment methods.

The work [70] considers facial alignment as a Nonlinear Least Squares problem and adapts Newton's to solve it. Using the training data, they learn a series of parameter updates, minimizing the objective. These updates are decomposed into the person-specific components and generic descent directions. The method learns average descent

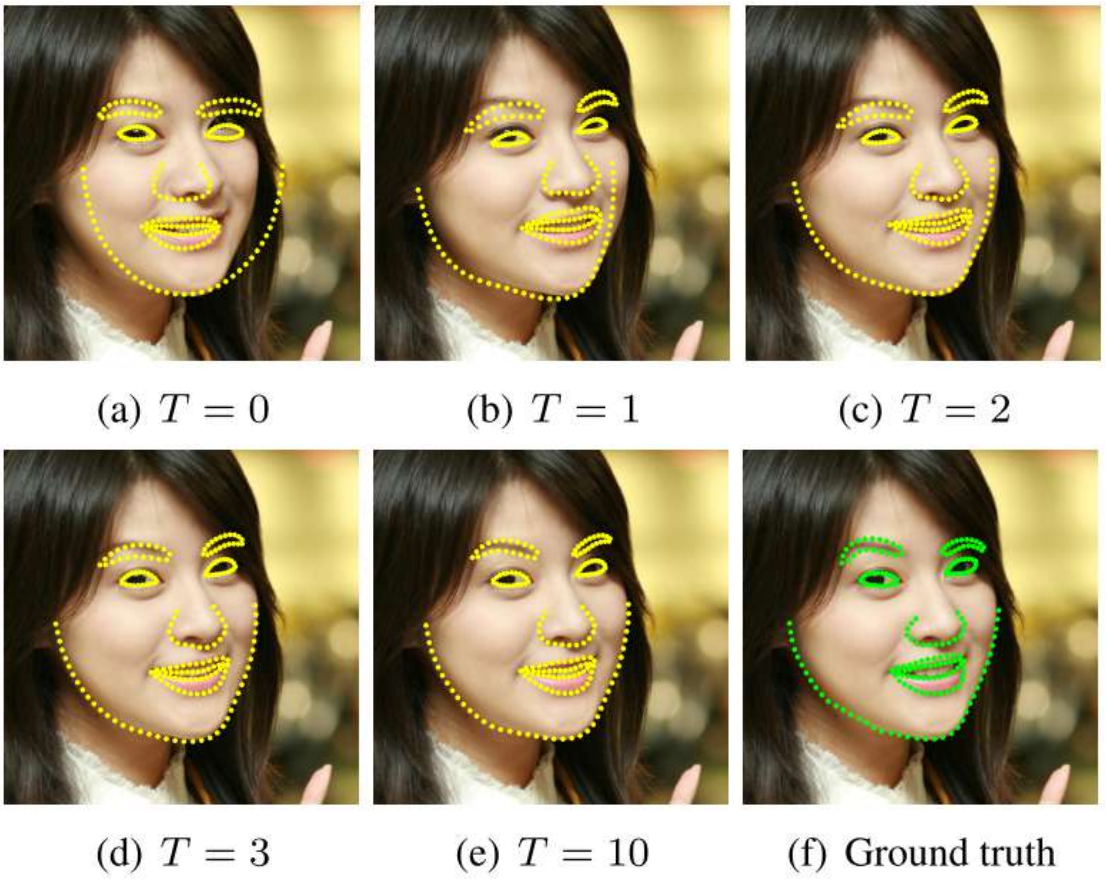


FIGURE 2.9: Landmark estimates at different levels of the cascade. Figure taken from [15].

directions during training. In testing, given an image of unseen person, an update is generated by projecting person-specific components onto the learned generic directions.

The work [15] also treats the task as a regression problem and suggests to track facial features with an ensemble of regression trees. Let $\mathbf{S} = (\mathbf{l}_1^T, \ldots, \mathbf{l}_p^T) \in \mathbb{R}^{2p}$ be coordinates of p facial landmarks. The regressors' cascade successively refines estimated coordinates (Figure 2.9):

$$\hat{\mathbf{S}}^{(t+1)} = \hat{\mathbf{S}}^{(t)} + r_t(I, \hat{\mathbf{S}}^{(t)}),$$

where I is an image, t and $(t+1)$ are indexes in the cascade, $r_t(I, \mathbf{S}$ is an update given by the regressor in the cascade. Initial shape is the mean landmark position along training data, which is centered and scaled based on the bounding box output of a face detector. During training, initial shape is picked up randomly from training data for data augmentation purposes. In order to have shape invariant split tests, during training, image is warped to the mean shape based on the current shape estimate at each iteration. A generalization to the case of missing labels is suggested. The approach works superior to real-time up to 1000 FPS.

The work [86] also adapts the cascaded pose regression. They learn the local binary features with random forest for each facial landmark independently. The obtained local binary features are used to jointly learn a linear regression for the final output.

The work [85] presents the Constrained Local Neural Field model for facial landmark detection. They proposed a probabilistic patch expert (landmark detector) that can learn non-linear and spatial relationships between the input pixels and the probability of a landmark being aligned. To fit the model a Non-uniform Regularized Landmark Mean-Shift optimization technique which takes into account the reliabilities of each patch expert was used.

2.4 General system for gaze redirection

Although suggested system differs in many aspects, such as the training data, actual type of predictor used, etc., there are number of standard steps, which all of the systems follow when processing the new face image. For the details, which are specific for different systems (different predictors), I refer reader to the comparison table 1.2 and to the following chapters. This is a concretization of the general pipeline for image re-synthesis Section 2.1.1 in application to a gaze redirection problem.

1. Having face image as the input, first a facial alignment software is applied (currently I use [84, 85]). It outputs facial landmarks (the example is in Figure 2.9), from which landmarks $\mathbf{l}_l$ and $\mathbf{l}_r$ are chosen, corresponding to the left and right eyes. If there are more than one person at the image (and all systems rely on facial landmark software to detect all faces), the next steps are applied successively to all of them.

2. Eyes are cropped using the eye localization described in Section 2.3. Right eye is mirrored, and all the next steps up to blending eyes back to the output image are applied for both eyes independently.

3. Depending on the type of predictor, eye image I and its landmarks $\mathbf{l}$ are possibly downsampled to the fixed scale, on which the predictor was trained. This operation is needed for predictors based on Neural Network, while Random Forest could process image on the native resolution.

4. Flow field is predicted
$$\mathcal{F} = p(I, \mathbf{l}).$$

The procedure here is specific for each system, because different systems have different inputs (possibly redirection angle or analogy pairs of images are additional inputs).

5. Depending on the type of predictor, warping field is possibly upsampled using bilinear interpolation (2.6) (again, for predictors based on Neural Networks).

6. The flow field $\mathcal{F}$ is linearly decreased to zeros near the border. Typically, for image size about 80×100 the border layer of thickness 5 is changed.

7. The output image of the eye is obtained, applying the flow field on the native resolution:
$$O = \mathbf{S}(I, \mathcal{F}).$$

8. Right eye is mirrored back, and both eye are pasted back to get the output image. As eye images are cropped from the input face images, in order to blend the result back without stitching artifacts, the flow maps are linearly decreased to zeros near the border of the cropped eye.

The procedure above outlines the general approach, however it does not address the way how the flow field $\mathcal{F} = p(I, \mathbf{l})$ is predicted. The next chapters will describe several systems that detail and expand on the general approach outlined in this section. These systems utilize different methods to learn the predictor from the dataset.

2.4.1 Assessment methods

One of the main components of developing methods for predicting the warping field is the assessment of the results. The assessment of the results of image re-synthesis should address two questions:

1. Was the image transformed in an intended way? In the context of gaze redirection, it means whether the gaze was redirected by a desired angle.

2. Is re-synthesized image photorealistic?

This work suggests three quantitative assessment methods. All of them evaluate the output image, i.e. the result of warping using the predicted warping field.

The first method is Mean Squared Error between the output and the ground truth image. The MSE measure is the most natural and easily estimated assessment method. The closeness to the ground truth intends the positive answer to both questions. However,

the MSE is an integral characteristic of the image, therefore, it could be not sensitive enough to some local changes. These local changes could both spoil the effect of the intended transformation and introduce artifacts, which decrease the photorealism of the result.

In order to measure whether the gaze was redirected by a desired angle, the additional model, which estimates the gaze difference between two images, is trained. As this model could itself be not perfect, the following assessment protocol is used: model error distribution of re-synthesized images is compared with model error distribution of ground-truth images for some fixed redirection angle. This assessment method answers only the first question because the trained model could be invariant to some non-realism in the images. The details on the method could be find in Section 5.3.2.

As the photorealism of the result is finally a subjective opinion of a user, the user study is suggested to answer the second question. Users could be asked in a different manner, possibly the simplest is to show them an image and to ask, whether it is photo-realistic. In this work, it was chosen to show user four images, one of each is re-synthesized, and three other are ground-truth and ask to find the one, which seems the most unrealistic. Analyzing the guess ratio, some conclusions about photorealism of the results could be drawn. The best guess ratio should be 25% – the performance of the random guess. Typically, users tend to determine the synthesized image more often. This assessment method does not address the question, whether the gaze was redirected by a desired angle, because a trivial method that produces zero warping field would get the perfect score. The details on the method and its two variants could be found in Section 4.6.2.1 and Section 5.3.4.

The experiments in Section 4.6 and Section 5.3 showed the strong correlation between the MSE measure and two other assessment methods, as well as with qualitative assessment of the results. Therefore, MSE measure is considered to be the most universal and is used in all quantitative experiments in the first place.

Chapter 3

Weakly-supervised random forest for image re-synthesis

Here, I present a new system for learning a warping field predictor for image re-synthesis. The synthesis is based on supervised machine learning, for which a large number of image pairs describing the the target transformation is collected and processed. The more advanced version of the system is based on a special kind of randomized decision tree ensembles called *warping flow forests*. The current implementation runs in real-time on a single core of a laptop with a clear potential for a real-time performance on tablet computers.

The system is illustrated and evaluated for a task of monocular gaze correction in video-conference. The system synthesizes realistic views with a gaze systematically redirected upwards (10 or 15 degrees in my experiments). At test-time, the system accomplishes gaze correction using simple pixel replacement operations that are localized to the vicinity of person's eyes, thus achieving high computational efficiency.

3.1 Overview of Random Forests

A data sample is denoted as $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$, x_i represent features. Their nature depends on application. For example, the usual case in computer vision is when $\mathbf{x}$ stands for some pixel p, and features x_i – for chosen pixels in the particular neighborhood of p. For example, x_i could be difference in intensities in a chosen channel between some pixel from neighborhood and pixel p.

A decision tree is a tree graph (graph without cycles), whose nodes are divided into two groups: split nodes and leaf nodes. Split nodes are all internal (not leaf) nodes of the

graph. Each split node j is associated with a split (or test) function:

$$h(\mathbf{x}, \theta_j) : \mathbb{R}^d \times \mathcal{T} \to \{0, 1\}, \tag{3.1}$$

where $\theta_j \in \mathcal{T}$ are the parameters of node j, $\mathcal{T}$ is parameter space. For each data point, tree is traversed from the root to leaf, and at each split node result of a test function determines, whether the data point proceeds to the left or to the right child of this split node. The common choice in random forest applications are decision stumps – data is splitted by thresholding one of the features:

$$h(\mathbf{x}; i, \tau) = [x_i < \tau]. \tag{3.2}$$

In a supervised scenario a training point $\mathbf{s}$ is a pair $(\mathbf{x}, \mathbf{y})$ of a data sample and some ground truth label, $\mathbf{y} \in \mathcal{Y}$. For example, $\mathcal{Y} = \{0, 1\}$ for a binary classification task, or $\mathcal{Y} = \mathbb{R}^n$ for a multivariate regression problem. In unsupervised scenario a training point $\mathbf{s} = \mathbf{x}$. I also write (3.1) as $h(\mathbf{s}, \theta_j)$, meaning that either $\mathbf{s} = (\mathbf{x}, \mathbf{y})$ or $\mathbf{s} = \mathbf{x}$ and $h(\mathbf{s}, \theta_j) = h(\mathbf{x}, \theta_j)$. A set of training points in the root of the tree is denoted as $\mathcal{S}_0$. In each split node j an incoming set $\mathcal{S}_j$ is recursively divided:

$$\mathcal{S}_j = \mathcal{S}_j^L \cup \mathcal{S}_j^L,$$
$$\mathcal{S}_j^L = \{\mathbf{s} \in \mathcal{S}_j |\ h(\mathbf{s}, \theta_j) = 0\}, \quad \mathcal{S}_j^R = \{\mathbf{s} \in \mathcal{S}_j |\ h(\mathbf{s}, \theta_j) = 1\}.$$

Left and right subsets are incoming for new nodes: $\mathcal{S}_j^L = S_{2j+1}$, $\mathcal{S}_j^R = S_{2j+2}$.

In test time new query sample $\mathbf{x}$ is traversed in the same recursive way, until it reaches a leaf node. Each leaf node contains a predictor, which associates a sample $\mathbf{x}$ with a target label $\mathbf{y} \in \mathcal{Y}$ (3.9).

3.1.1 Training a Decision Tree

Learning of a split function (3.1), e.g. assigning some value to a set of parameters θ_j is a core question in applying decision trees. It is often done by optimizing some objective function:

$$\theta_j = \arg\max_{\theta \in \mathcal{T}} I(\mathcal{S}_j, \theta). \tag{3.3}$$

The optimization and splitting of the initial training set is often performed in a greed recursive manner from root to leaves. Training begins at the root node, $j = 0$. A training set is divided into two disjoint subsets in the two child nodes. Recursively applying this procedure to all successively constructed nodes, we continue the training phase until a

stopping criterion is met. Different stopping criteria are possible. The most common are:

- a maximum depth of the tree D has been reached;
- some minimum threshold value of the information is gained (3.4), which means that intended attributes of training point are the same across all samples in current node, or at least dividing samples in current node into two new sets does not give enough information gain;
- current node contains some minimum threshold amount of samples.

The common choice of an objective function in (3.3) is an information gain from splitting a node into two new nodes:

$$I = H(\mathcal{S}_j) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i). \tag{3.4}$$

Here $H(\mathcal{S})$ is some measure of compatibility of set $\mathcal{S}$. For example, Shannon entropy is a possible choice for a classification problem:

$$H(\mathcal{S}) = -\sum_{c \in C} p(c) log(p(c)). \tag{3.5}$$

Here C is the set of all classes, and $p(c)$ is the empirical distribution over classes in the set $\mathcal{S}$. Another popular choice of $H(\mathcal{S})$ in classification problems is Gini impurity:

$$H(\mathcal{S}) = \sum_{c \in C} p(c) 2 * (1 - (p(c)). \tag{3.6}$$

The standard choice for regression problem is error of fit:

$$H(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{s} \in \mathcal{S}} (y - \bar{y})^2, \tag{3.7}$$

where $\bar{y}$ is the mean of all points in the node.

After learning phase in supervised problem, each leaf node l contains a subset of labeled training data. In general cases the leaf statistics defines a distribution

$$p_l(y|\mathbf{x}), \tag{3.8}$$

which can be, for cases of classification and regression, conditional distribution of categorical label $p(c|\mathbf{x})$ or continuous label $p(y|\mathbf{x})$. During testing a query point is traversed through the tree until the leaf is reached. The tree is constructed in such a way, that the

new query point would probably finish up in a leaf, containing similar training points. Thus, the reasonable assumption for labeling of this new test point is to put label, similar to the training points in that leaf. For example, Maximium-A-Posteriori of the leaf label statistics could be used:

$$\hat{y} = arg\max_{y} p(y|\mathbf{x}). \tag{3.9}$$

In case of classification it corresponds to picking a class with a highest frequency, in case of regression – mean of all training points, falling into particular leaf.

3.1.2 Random Forests

Random Forest is an ensemble of several Decision Trees. It was introduced in [87, 88] Randomness could be injected in two main ways:

- random sampling of training set for each tree in forest (e.g. bagging);
- random optimizing of splits (3.3).

Bagging was introduced in [88, 89] as a method to reduce overfitting. Each tree in forest is trained on a random subset of the whole training set. Such method increases generalization, avoiding all trees from specialization to a single dataset. Training of each tree also becomes faster with this strategy.

The optimization problem written in form (3.3) is rarely tractable in practice, because space of all possible parameters $\mathcal{T}$ is very large. Thus each split is optimized over some small random subset of parameters $\mathcal{T}_j \subset \mathcal{T}$.

$$\theta_j = \arg\max_{\theta \in \mathcal{T}_j} I(\mathcal{S}_j, \theta).$$

Typically, the optimization is done via brute force search over the set $\mathcal{T}_j$. For a typical case of decision stumps (3.2), one could wish to randomize either both feature number i and threshold τ, or only one of them. For example, in work [90] some small numbers of features is considered for each split train. All subset of samples in the node is sorted according to the chosen feature value, and thus an optimal value of threshold is found linearly of the subsample size.

More detailed definition of abstract Decision Tree and Random Forest models could be found in [91].

3.1.3 Application of Random Forests in Computer Vision

The variant of the gaze correction system presented in this chapter continues the long line of real-time computer vision systems based on randomized decision trees [87, 88] that includes real-time object pose estimation [92], head pose estimation [93], human body pose estimation [16], background segmentation [94], etc.

Authors of work [92] consider such problems as object detection and pose estimation. They match keypoints from the input image with the ones on a target object. They formulate the point matching as classification problem and use random forests to solve it. Formally, at train time they select a set $\mathcal{K}$ of cardinality K of keypoints int he object model. At test-time the select a patch, centering it around a found keypoint. The task is to determine whether it matches one of K keypoints in $\mathcal{K}$ or not, i.e. solve a classification problem with $K + 1$ labels $\{-1, 1, 2, \ldots, K\}$, where -1 stands for a situation where a keypoint does not belong to $\mathcal{K}$. The suggested difference of pixels intensities taken in the neighborhood of the keypoint as a binary node tests (3.1) in classification trees. They compare two techniques for growing a tree. The first one is an approach described above: picking of several tests at random and greed choosing of the best among them according to the information gain (3.4). The number of picked up tests was 10 at the root node and $100d$ for a node at depth d, $d \geq 2$. The second is called extremely randomized trees. In this approach, only one test is picked up at random, without any optimization at all. This approach significantly decreases training time with a small loss of quality.

In paper [16] authors suggest two approaches for human body pose estimation: body part classification (BPC) and offset joint regression (OJR). Both approaches make use of random forest, which is applied individually to each pixel of a depth image. BPC predicts body parts labels for each pixel of the image. A density over 3D world space could be calculated by reprojecting per-pixel label probabilities, knowing the depth image and calibration of the depth camera. OJR instead directly predicts a set of 3D offset votes from each pixel to the body joints using a regression forest Figure 3.1. The split tests employed at internal nodes (3.1) are differences between two pixels in the neighborhood of a query pixel. 2D offsets are normalized by the depth value to make features depth invariant. Authors choose approximately half of tests in such a way, that one of two compared pixels is a query pixel itself.

While classical random forests are trained to do classification or regression, the trees in my method predict some (unnormalized) distributions. My method is thus related to other methods that use structured-output random forests (e.g. [90, 95]).

A Hough-transform-based object detection method is suggested in work [95]. They suggest to train a forest to map the appearance of an image patch into its Hough vote,

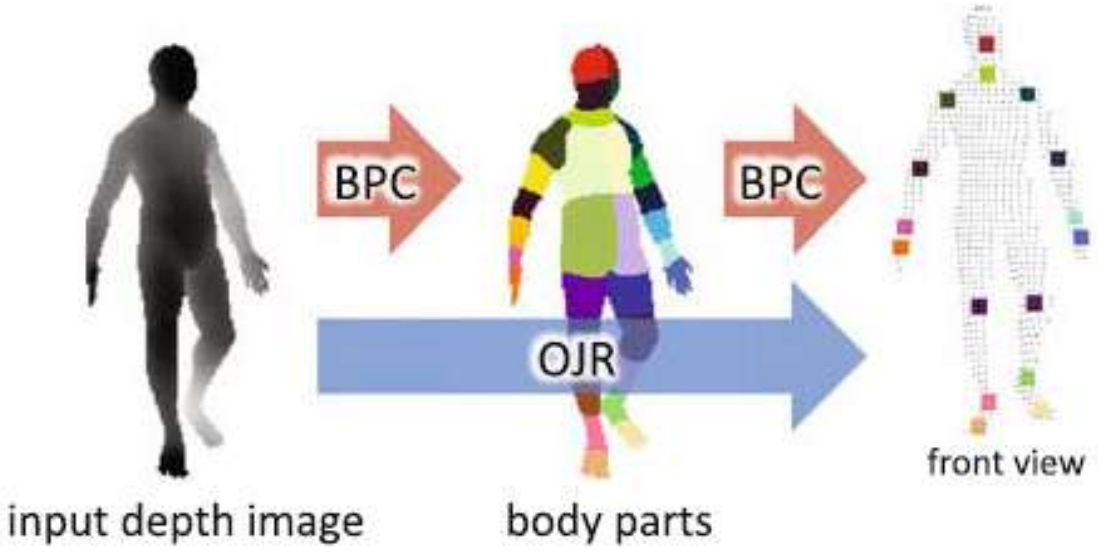


FIGURE 3.1: Body part classification (BPC) first predicts a (color-coded) body part label at each pixel, and then uses these inferred labels to localize the body joints. Offset joint regression (OJR) instead directly regresses the joint positions. Figure is taken from [16]

.

incorporating it in the decision forest framework. Object detection problem is decomposed into classifying patches belonging to an object or a background and regressing the offset vector of the patch $\mathbf{v}$ to the centroid of the object. Thus, the leaf statistics (3.8) in Hough forests capture both categorical class label $p(c|\mathbf{v})$ and continuous offset $p(\mathbf{y}|c, \mathbf{v})$. However, they are not independent: the continuous offset $\mathbf{y}$ depends on the categorical label c. Unifying two distributions, the statistics on the leaf will show the probability of an object of a particular class c to appear in a patch with appearance $\mathbf{v}$ at the relative location $\mathbf{y}$:

$$p(\mathbf{y}, c|\mathbf{v}) = p(\mathbf{y}|c, \mathbf{v})p(c|\mathbf{v}).$$

Both classification (3.5) and regression (3.7) measures of compatibility are used. At each node, a random decision is made whether to optimize split based on one or another with equal probabilities.

The work [90] applied structured random forests to edge detection. In their approach, $\mathbf{y} \in \mathcal{Y}$ is some structured image annotation, e.g. edge mask. The idea is to map all examples fallen in the given node to a discrete set of labels $\mathcal{C} = \{1, \ldots, k\}$, in such a way that similar labels $\mathbf{y} \in \mathcal{Y}$ are mapped to the same label. After that, the standard information criterion for classification ((3.5), (3.6)) could be used to train a split.

Firstly [90] uses intermediate mapping $\Pi : \mathcal{Y} \to \mathcal{Z}$ to some Euclidean space $\mathcal{Z}$. Then they use some clusterization technique to to obtain discrete labels $\mathcal{C} = \{1, \ldots, k\}$. In

application to edge detection, $\mathbf{y} \in \mathcal{Y}$ are 16×16 segmentation masks and $z \in \mathcal{Z}$ are binary vectors, defining whether each pair of pixel is in the same or different vectors. In practice, they sample only subset of $m = 256$ dimensions of this binary vectors, and further reduce to 5 dimensions by PCA.

Their ensemble model of combining a set of n labels $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathcal{Y}$ into a single prediction is again to use mapping $\Pi : \mathcal{Y} \to \mathcal{Z}$ to go to Euclidean space and then to choose a medoid sample, i.e. among $z_k = \Pi(y_k)$ they choose such k, that z_k gives the minimum to the sum of distances to all other z_i. The ensemble model is necessary for both setting the label in the leaf node and uniting prediction from several trees into the single one.

The trees in my method are trained under weak supervision, and this relates my work to such methods as [76]. The suggested filter forests learns to assign optimal filters $\mathbf{w}$ to inputs $\mathbf{x}_i \in \mathcal{X}$ to learn some mapping $f_{\mathbf{w}} : \mathcal{X} \to \mathcal{Y}$, $f_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{x}_i^T \mathbf{w}$. The forest is weakly-supervised in the sense that the training data $(\mathbf{x}_i, \mathbf{y}_i)$ (where $\mathbf{y}_i \in \mathcal{Y}$ is the appearance of the patch after filter, e.g. denoised patch) does not contain target filters. They define compatibility measure as

$$H(\mathcal{S}) = ||\mathbf{Y} - \mathbf{X}\mathbf{w}^*||^2,$$

where $(\mathbf{X}, \mathbf{Y})$ consists of all samples in the node, and $\mathbf{w}^*$ is the optimal filter in the node, which is learned using the same ℓ_2 loss with some data-dependent regularizing term. The similar approach for learning optimal warping flow vectors is applied in the method suggested in this paper (Section 3.2.3). However, in addition to this idea, in the context of weak supervision the non-standard way of ensembling the trees is suggested (Section 3.2.2).

3.2 Warping flow forest

As described in Section 2.1 the suggested approach is to predict the warping field for the input image. As we do not have the warping field (2.4) for the training pairs, at this section I describe the system based on *weakly-supervised* random forests (*warping flow forests*).

3.2.1 Image-independent warping field

Under my approach, I can propose a very simple baseline that suggests a fixed warping field (2.4), i.e. independent of the test image content and based solely on the relative position in the estimated bounding box, i.e. $u = u(x/\Delta, y/\Delta)$ and $v = v(x/\Delta, y/\Delta)$,

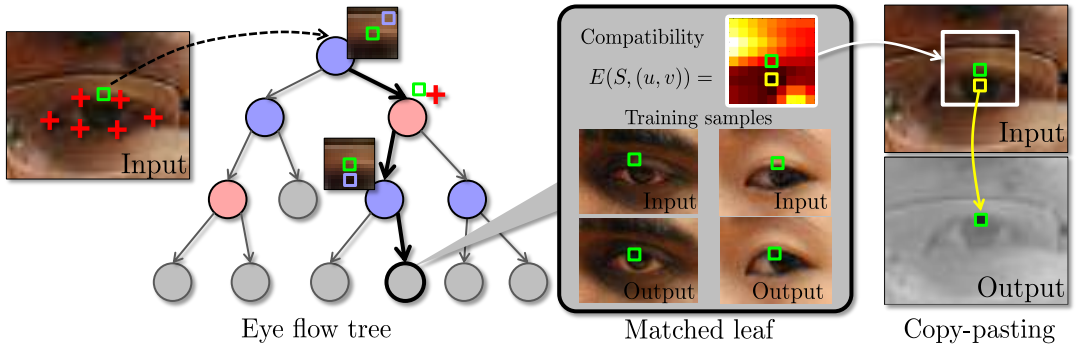


FIGURE 3.2: **Processing of a pixel (green square) at test time in an warping flow tree.** The pixel is passed through an warping flow tree by applying a sequence of tests that compare the position of the pixels w.r.t. the feature points (red crosses) or compare the differences in intensity with adjacent pixels (bluish squares) with some threshold. Once a leaf is reached, this leaf defines a matching of an input pixel with other pixels in the training data. The leaf stores the map of the compatibilities between such pixels and warping flow vectors. The system then takes the optimal warping flow vector (yellow square minus green square) and uses it to copy-paste an appropriately-displaced pixel in place of the input pixel into the output image. Here, a one tree version is shown for clarity, the actual system would sum up the compatibility scores coming from several trees before making the decision about the warping flow vector to use.

where the values of u and v for a given relative location $(x/\Delta, y/\Delta)$ are learned on training data as discussed below.

3.2.2 Architecture of warping flow forest

At test time, this system matches a pixel at (x, y) to a group of similar pixels in training data and finds the most appropriate warping flow vector for this kind of pixels. To achieve this effect, a pixel is passed through a set of specially-trained ensemble of randomized decision trees (*warping flow trees*). When a pixel (x, y) is passed through an warping flow tree, a sequence of simple tests of two kinds are applied to it. A test of the first kind (*an appearance test*) is determined by a small displacement (dx, dy), a color channel $c \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$, and a threshold τ and compares the difference of two pixel values in that color channel with the threshold:

$$I(x + dx, y + dy)[c] - I(x, y)[c] \gtrless \tau \tag{3.10}$$

A test of the second kind (*a location test*) is determined by the number of the feature point $i \in \{1, \ldots N\}$, $\mathbf{l}_i = (l_i^x, l_i^y)$ and a threshold τ and compares either $x - l_i^x$ or $y - l_i^y$ with τ:

$$x - l_i^x \gtrless \tau \qquad y - l_i^y \gtrless \tau \tag{3.11}$$

Through the sequence of tests, the tree is traversed till a leaf node is reached. Given an ensemble of T warping flow trees, a pixel is thus matched to T leaves.

Each of the leaves contain an unnormalized distribution of compatibility score (3.12) over the warping flow vectors (u, v) for the training examples that fell into that leaf at learning stage Figure 3.2. I then sum the T distributions corresponding to T leaves, and pick (u, v) that minimizes the aggregated distribution. This (u, v) is used for the copy-paste operation (2.2).

Handling scale variations. To make matching and replacement operations covariant with the changes of scale, a special care has to be taken. For this, I rescale all training samples to have the same characteristic radius Δ_0. During gaze redirection at test time, for an eye with the characteristic radius Δ I work at the native resolution of the input image. However, when descending an warping flow tree, I multiply the displacements (dx, dy) in (3.10) and the τ value in (3.11) by the ratio Δ/Δ_0. Likewise, during copy-paste operations, I multiply the warping flow vector (u, v) taken from the image-independent field or inferred by the forest by the same ratio. To avoid the time-consuming interpolation operations, all values (except for τ) are rounded to the nearest integer after the multiplication.

3.2.3 Learning

I assume that a set of training image pairs (I^j, O^j) is given. I assume that within each pair, the images correspond to the same head pose of the same person, same imaging conditions, etc., and differ only in the gaze direction (Figure 2.8). I further assume that the difference in gaze direction is the same for all training pairs (separate predictor needs to be trained for every angular difference). As discussed above, I also rescale all pairs based on the characteristic radius of the eye in the *input* image.

For each pixel (x, y), our goal is to turn the color of the pixel $I^j(x, y)$ into the color given by $O^j(x, y)$ by applying the operation (2.2). Therefore, each pixel (x, y) within the bounding box B specifies a training tuple $\mathcal{S} = \{(x, y), I, \{\mathbf{l}_i\}, O(x, y)\}$, which includes the position (x, y) of the pixel, the input image I it is sampled from, eye feature points $\{\mathbf{l}_i)\}$ in the input image, and finally the color $O(x, y)$ of the pixel in the output image. The trees or the image-independent flow field are then trained based on the sets of the training tuples (training samples).

As discussed above, unlike most other decision trees, warping flow trees have to be trained in a weakly-supervised manner. This is because each training sample does not include the target flow field $\mathcal{F}(x, y) = (u(x, y), v(x, y))$ that the tree is designed to

predict. Instead, only the desired output color $O(x, y)$ is known, while same colors can often be obtained through different offsets and adjustments making the supervision "weak".

The goal of the training is then to build a tree that splits the space of training examples into regions, so that for each region replacement (2.2) with the same warping flow vector (u, v) produces good result for all training samples that fall into that region. Given a set of training samples $\mathbf{S} = \{\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^K\}$, I define the *compatibility score* H of this set with the warping flow (u, v) in the following natural way:

$$H\left(\mathbf{S}, (u, v)\right) = \sum_{k=1}^{K} \sum_{c=R,G,B} \left| I^k(x^k + u, y^k + v)[c] - O^k(x^k, y^k)[c] \right| . \tag{3.12}$$

Here, the superscript ^{k} denotes the characteristics of the training sample $\mathcal{S}^k$, I^k and O^k denote the input and the output images corresponding to the kth training sample in the group $\mathbf{S}$, and c iterates over color channels. Overall, the compatibility $E(\mathbf{S}, (u, v))$ measures the disparity between the target colors $O^k(x^k, y^k)$ and the colors that the replacement process (2.2) produces.

Given the compatibility score (3.12) I define the *coherence score* $\tilde{H}$ of a set of training samples $\mathbf{S} = \{\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^K\}$ as:

$$\tilde{H}(\mathbf{S}) = \min_{(u,v)\in\mathcal{Q}} H\left(\mathbf{S}, (u, v)\right) , \tag{3.13}$$

Here, $\mathcal{Q}$ denotes the search range for (u, v), which is taken to be a square $[-R, \ldots R] \otimes [-R, \ldots R]$ sampled at integer points. Overall, the coherence score is small as long as the set of training examples is compatible with some warping flow vector $(u, v) \in \mathcal{Q}$, i.e. replacement (2.2) with this flow vector produces colors that are similar to the desired ones.

The coherence score (3.13) then allows us to proceed with the top-down growing of the tree. As is done commonly, the construction of the tree proceeds recursively. At each step, given a set of training samples $\mathbf{S}$, a large number of tests (3.10),(3.11) are sampled. Each test is then applied to all samples in the group, thus splitting $\mathbf{S}$ into two subgroups $\mathbf{S}_1$ and $\mathbf{S}_2$. A quality of the split $(\mathbf{S}_1, \mathbf{S}_2)$ is then defined as:

$$I(\mathbf{S}_1, \mathbf{S}_2) = \frac{|\mathbf{S}_1|}{|\mathbf{S}|}\tilde{H}(\mathbf{S}_1) + \frac{|\mathbf{S}_2|}{|\mathbf{S}|}\tilde{H}(\mathbf{S}_2) + \lambda\big||\mathbf{S}_1| - |\mathbf{S}_2|\big| , \tag{3.14}$$

where the last term penalizes the unbalanced splits proportionally to the difference in the size of the subgroups. This term typically guides the learning through the initial stages near the top of the tree, when the coherence scores (3.13) are all "bad" and becomes

relatively less important towards the leaves. After all generated tests are scored using (3.14), the test that has the best (minimal) score is chosen and the corresponding node V is inserted into the tree. The construction procedure then recurses to the sets $\mathbf{S}_1$ and $\mathbf{S}_2$ associated with the selected test, and the resulting nodes become the children of V in the tree.

The recursion stops when the size of the training sample set $\mathbf{S}$ reaching the node falls below the threshold τ_S or the coherence $\tilde{H}(\mathbf{S})$ of this set falls below the threshold τ_C, at which point a leaf node is created. In this leaf node, the compatibility scores $E(\mathbf{S}, (u, v))$ for all (u, v) from $\mathcal{Q}$ are recorded. As is done conventionally, different trees in the ensemble are trained on random subsets of the training data, which increases randomization between the obtained trees.

Learning the image-independent warping field is much easier than training warping flow trees. For this, I consider all training examples for a given location (x, y) and evaluate the compatibility scores (3.12) for every offset $(u, v) \in \mathcal{Q}$. The offset minimizing the compatibility score is then recorded into the field for the given (x, y).

Discussion of the learning. By predicting the warping flow $(u(x, y), v(x, y))$ I do not aim to recover the apparent motion of a pixel (x, y). Indeed, while recovering the apparent motion might be possible for some pixels, apparent motion vectors are not defined for dis-occluded pixels, which inevitably appear due to the relative motion of an eyeball and a lower eyelid. Instead, the learned predictors simply exploit statistical dependencies between the pixels in the input and the output images. As is demonstrated in Section 3.3, recovering such dependencies using discriminative learning and exploiting them allows to produce sufficiently realistic emulations of gaze redirection.

3.2.4 Implementation details

Learning the forest. When learning each split in a node of a tree, I first draw randomly several tests without specifying thresholds. Namely, for each test I first randomly sample a type of the test, choosing between the appearance test and the location test with equal probability. I then sample parameters of test uniformly from a certain range. Thus I sample dx, dy (from the 9×9 neighborhood) and a channel c for appearance tests (3.10), or the number of the feature point for location tests (3.11). I then learn an optimal threshold for each of the drawn test. In more detail, denote as h the left-hand-sides of expressions (3.10), (3.11), and $h_1, \ldots, h_K$ — all the data sorted by this expression. I then sort all thresholds of the form $\frac{h_i + h_{i+1}}{2}$ and probe them one-by-one (using an efficient update of the coherence scores (3.13) and the quality score (3.14) as

inspired by [90]). I set the probabilities of choosing a test for split in such a way that approximately half of tests are appearance tests and half are location tests.

To randomize the trees and speed up training, I learn each tree on random part of the data. Afterwards I "repopulate" each tree using the whole training data, i.e. I pass all training samples through the tree and update the distributions of replacement error in the leaves. Thus, the structure of each tree is learned on random part of the data but the leaves contain the error distribution of all data.

After picking up and summing of unnormalized compatibility score distributions from T trees, a minimizing flow vector could be found with a sub-pixel accuracy [96]. The distribution in the neighborhood of the minimizing pixel is approximated with a parabola using values in the pixel and its 4 neighbours. The minimum of the parabola is thus the estimated minimum of the distribution with a sub-pixel accuracy. If i is x-coordinate of the minimum of the discrete distribution $H(x)$ and $H(i-1) \neq H(i+1)$ (otherwise minimum is exactly at $x = i$), then the coordinate of the sub-pixel minimum is

$$x^* = i + \frac{H(i-1) - H(i+1)}{2(H(i-1) + H(i+1) - 2H(i))}, \tag{3.15}$$

and similarly for y-coordinate.

The warping flow forest system is trained for a specific angular difference. If needed, multiple separate models for different discretized angular differences could be trained. For example, for a video conference setup one can use $10, 15, 20, 25, 30$ degrees depending on the distance between the face and the screen. However, I found that the $15°$ vertical redirection produce convincing results for a typical distance between a person and a laptop and a typical laptop sizes, so I focus on this angular difference in the experiments.

The images in the Columbia dataset [83] are only loosely registered. Therefore, to perfectly match cropped eyes, I apply multistage registration procedure to images from this database. Firstly, before cropping an eye, I fit the similarity transformation based on all facial feature points except those corresponding to eyes. In the case of [70] there are totally 49 points and 6 of them corresponds to each eye, so I match similarity based on 37 points. I then crop roughly registered set of samples $\mathbf{S}_1 = (I_1^j, O_1^j)$, learn warping flow forest on $\mathbf{S}_1$, apply it to the set $\{I_1^j\}$ and get the resulting set $\{\hat{O}_1^j\}$. At the second stage I register images $\{O_1^j\}$ with $\{\hat{O}_1^j\}$ by translation (at one pixel resolution), by finding the shift that maximize the correlation of the Laplacian-filtered versions of the images. I exclude the dilated convex hull of eye features from the computation of the correlation, thus basing the alignment of external structures such as eye brows. I apply the optimal shifts to the output images in each pair, getting the second set of samples $\mathbf{S}_2 = (I_2^j = I_1^j, O_2^j)$.

At the final stage of registration I learn an warping flow forest $\mathbf{S}_2$, apply it to each of the input images $\{I_2^j\}$ and get the output image $\{\hat{O}_2^j\}$. I then register images $\{O_2^j\}$ and $\{\hat{O}_2^j\}$ in the same way as during the second stage, except that I do not exclude any pixels this time. This produces the final training set $\mathbf{S} = (I^j, O^j)$. At the end I manually throw away all training samples where the multistage registration failed.

Face registration slightly decreases both training and validation error on a Columbia dataset, where images are worse registered. However, this effect was not noticed on the Skoltech dataset, so face registration is not applied in the final variant of the method.

Numerical parameters. In the current implementation I rescale all cropped eyes to the resolution 50×40. I take the parameters of the bounding box $\alpha = 2.0$, $\beta = 1.6$, the parameter $\lambda = 10$, $R = 4$ and learn a forest of six trees. I stop learning splits and make a new leaf if one of the stopping criteria is satisfied: either coherence score (3.13) in the node is less than 1300 or the number of samples in the node is less than 128. Typically trees have around 2000 leaves and the depth around 12.

The bounding box parameters were chosen in such a way, that the whole region of the eye image, which is changing during the gaze redirection, is inside the bounding box, with several pixels width gap in reserve. The regularization weight λ and stopping parameters were optimized by training models for parameters values taken uniformly from some range. The criterion was a validation error, similar to the one presented in the experimental section (Figure 3.3). The size of the patch was chosen as a trade-off between the representation power of the model (resulting in lower validation error) and the memory required to store the model.

3.3 Experiments

In this section I provide computational experiments, illustrating the suggested approach for learning warping flow forest. Both quantitative and qualitative results are given. More results could be found in comparisons with other methods in the following chapters (Section 4.6 and Section 5.3).

Quantitative evaluation. I provide a quantitative assess of suggested method on the Columbia gaze dataset [83]. I sample image pairs applying the same preprocessing steps as when preparing data for learning. I make an eight-fold cross validation experiment: I split the initial set of 56 subjects, taking 7 as the test set and leaving 49 in the training set. I compare several methods that was applied to the input image of each pair, and then compute the mean square error (MSE) between the synthesized and the output images. I normalize the MSE error by the mean squared difference between the input

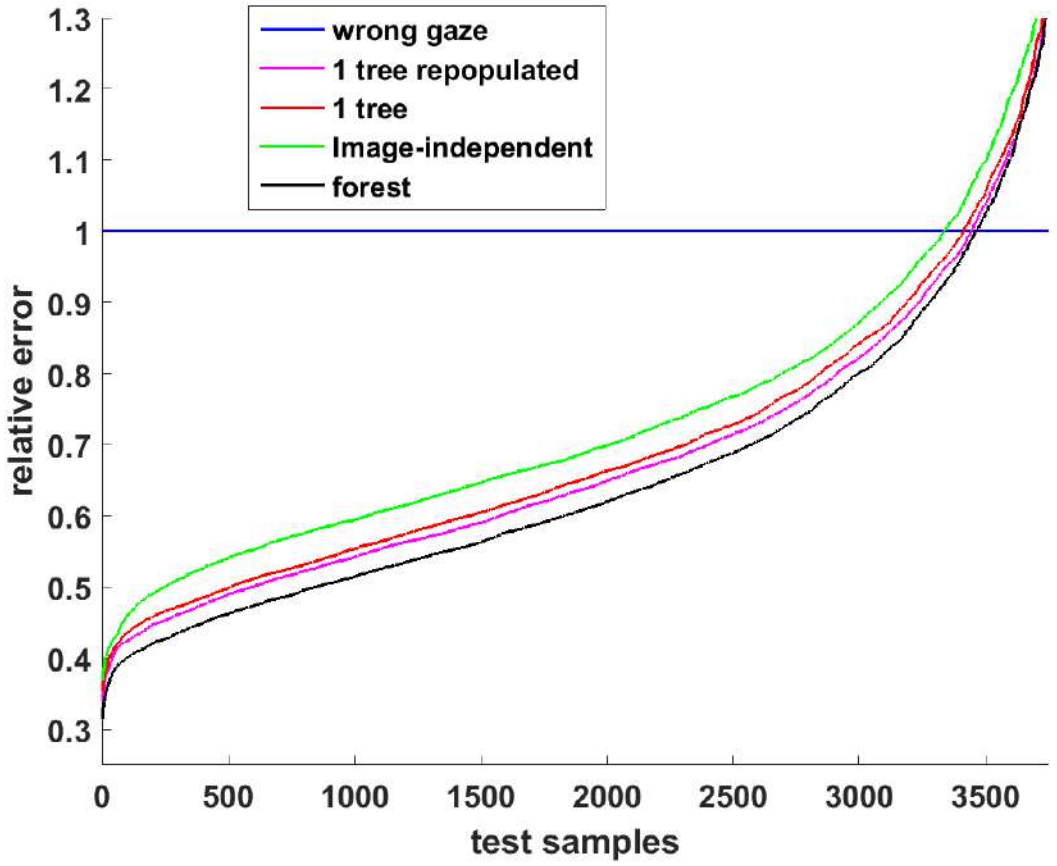


FIGURE 3.3: Quantitative evaluation on the testing sets in the eight-fold experiment: ordered normed MSE errors between the result and the "ground-truth". I compare the one tree version of suggested method (red), the six trees (black), one tree with repopulation (magenta), and the image-independent flow field version of the system (green). Increasing the tree number from six to ten or repopulating six trees does not give an improvement. For each method the normalized errors are sorted in the ascending order and then used to produce the curve.

and the output image. At each split i I compute the mean error e_i. To compare two methods, I evaluate the differences between their means and the standard deviation of these differences over the eight splits (Table 3.1).

For each method, I also sort the normalized errors in the ascending order and plot them on a graph (Figure 3.3). The quantitative evaluation shows the advantage of the tree-based versions of suggested method over the image-independent field variant. Full forest variants perform better than those based on a single tree. It is nevertheless interesting to see that the image-independent field performs well, thus verifying the general idea of attacking the gaze correction problem using a data-driven learning-based approach. Also, repopulation increases results for a single tree, but not for the full forest. Ten trees does not give significant improvement over six trees.

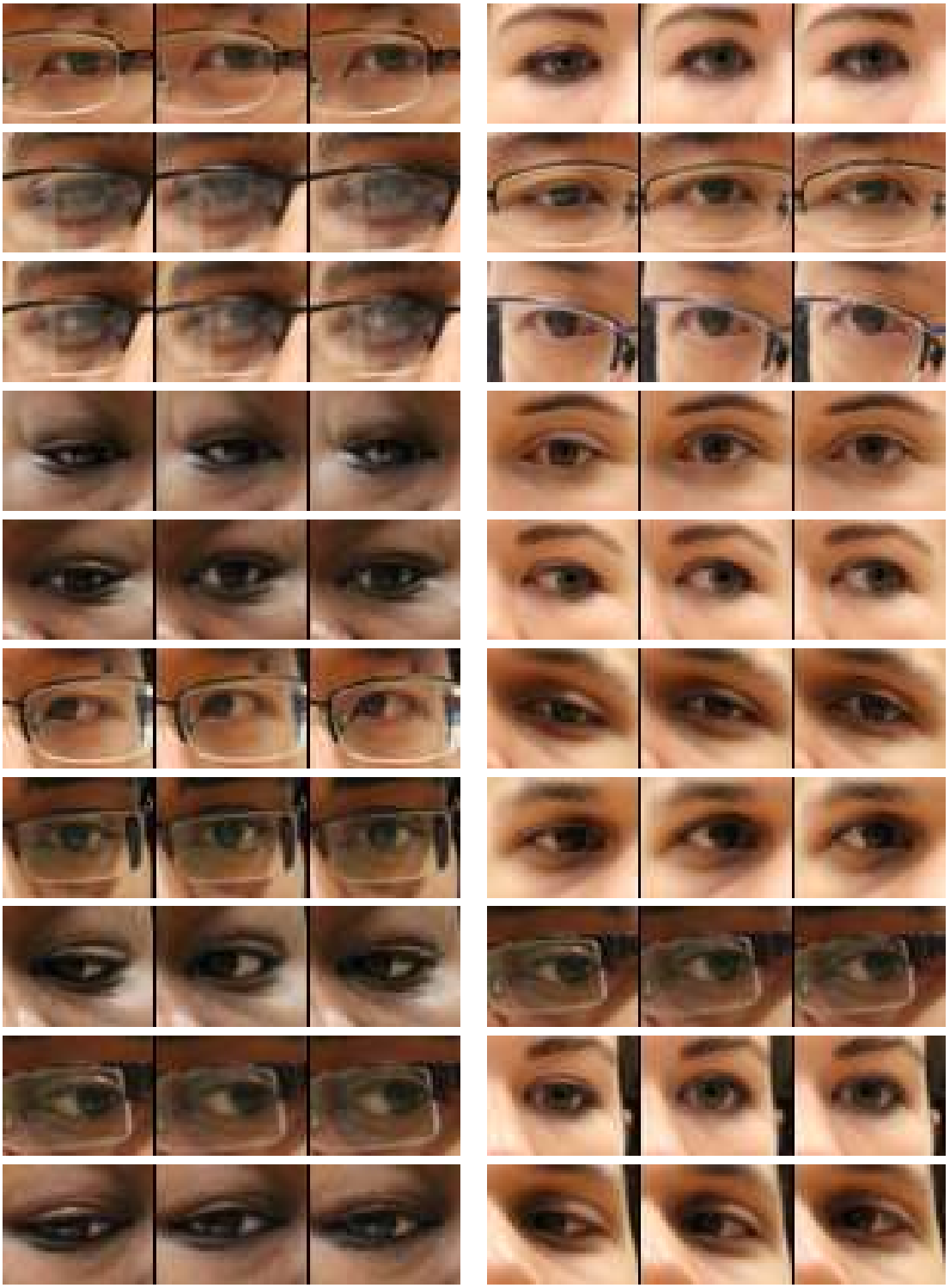


FIGURE 3.4: Randomly sampled results on the Columbia Gaze dataset. In each triplet, the left is the input, the middle example is the "ground truth" (same person looking 10 degrees higher). The right image is the output of warping flow forest. A stable performance of the method across demographics variations can be observed.

pair of methods	**mean**	σ
6 trees forest *vs* image independent	-0.079	0.0066
6 trees forest *vs* single tree	-0.038	0.0059
single tree repopulated *vs* single tree no repopulation	-0.015	0.0022
6 trees forest *vs* 10 trees forest	0.0036	0.0015
6 trees no repopulation *vs* 6 trees repopulated	0.00073	0.0017
single tree *vs* image independent	-0.055	0.0098

TABLE 3.1: The differences of mean errors between pairs of methods and standard deviations of these differences in the 8-fold cross validation test. Negative mean value means that first method in pair has a smaller mean error (works better).

Qualitative evaluation. Due to the nature of the application, the best way for the qualitative evaluation is watching the **supplementary video** at the project webpage [97] that demonstrates the operation of the method (six warping flow trees). Here, I also show the random subset of the results on the hold out part of the Columbia gaze dataset (10 degrees redirection) in Figure 3.4 and on the hold out part of the Skoltech gaze dataset (15 degrees redirection) in Figure 3.5. While the number of people in the training datasets was limited, one can observe that the system is able to learn to redirect the gaze of unseen people rather reliably obtaining a close match with the "ground truth" in the case of ten degree redirection (Figure 3.4).

One crucial type of failure is insufficient eye "expansion", which gives an impression of the gaze redirected on an angle less than required. Other types of artifacts include unstructured noise and structured mistakes on glass rims (which is partially explained by a small number of training examples with glasses in this split). Examples of failure cases are presented on Figure 3.6.

I further show the cutouts from the screenshots of the system (six warping flow trees) running live on a stream from a laptop webcam Figure 3.7. I use the same forest learned on the training part of the Columbia gaze dataset. Several sessions corresponding to different people of different demographics as well as different lighting conditions are represented.

Computation speed. The main testbed is a standard 640×480 streams from a laptop camera. On top of the feature tracking time, the forest-based version of the method requires between 3 to 30 ms to perform the remaining operations (querying the forest, picking optimal warping flow vectors and performing replacements). And the feature tracking time is very fast using modern methods, typically about 1 ms (Section 2.3.1).

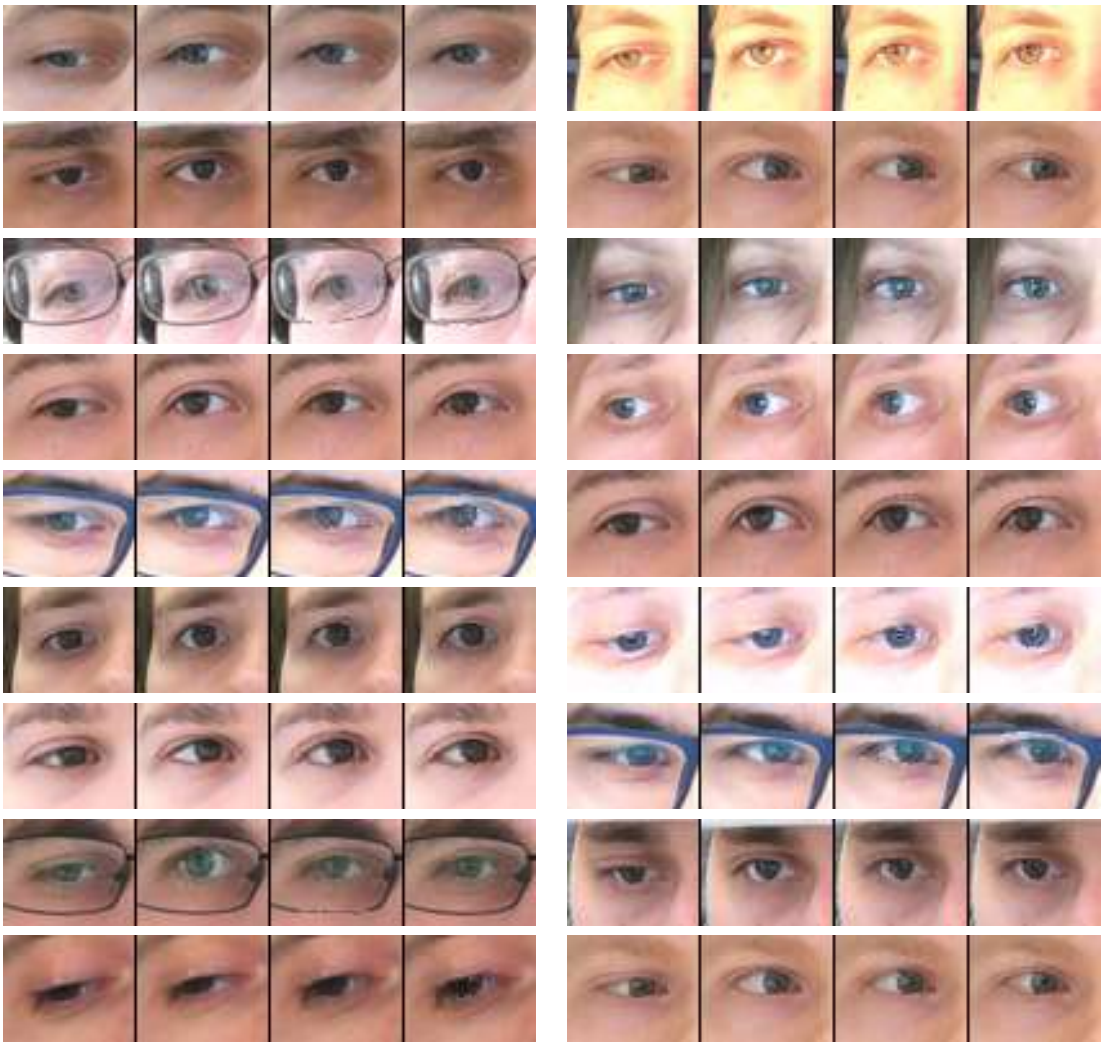


FIGURE 3.5: Randomly-sampled results on the Skoltech dataset (redirection by 15 degrees). In each tuple: (1) the input image, (2) the "ground truth", (3) the output of warping flow forest, (4) the output of image-independent field. Zoom-in recommended in order to assess the difference between the two variants of the method. The following types of failures are observed: insufficient eye expansion (bottom-left), noise artifacts, artifacts on glass rims (caused by a small number of people with glasses in the training set).

The large variability is due to the fact that the bulk of operation is linear in the number of pixels we need to process, so the 30 ms figure correspond to the situation with the face spanning the whole vertical dimension of the frame. Further trade-offs between the speed and the quality can be made if needed (e.g. reducing the number of trees from six to three will bring only very minor degradation in quality and almost a two-fold speedup).

Temporal stability. The temporal stability of the method could not be better than the temporal stability of the facial alignment method used because it affects both the

FIGURE 3.6: **Failure cases** on the Skoltech dataset (redirection by 15 degrees). In each quad: input image, the "ground truth", the output of warping flow forest, the output of image-independent field. The following types of failures are observed: insufficient eye expansion, noise artifacts, artifacts on glass rims.

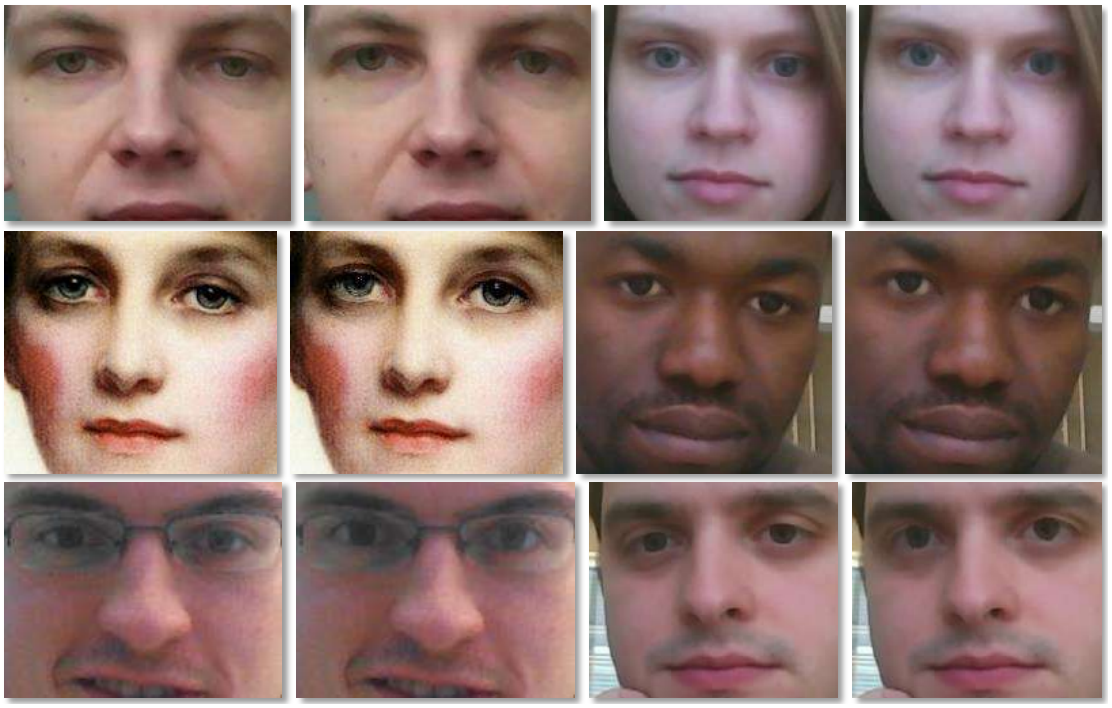


FIGURE 3.7: Qualitative examples of suggested system (based on six trees) showing the cutout of the frames of a video stream coming from a webcam (left – input, right – output). In the first two rows gaze is redirected by 10 degrees upwards, in the third — by 15 degrees. The system induces subtle changes that result in gaze redirection. Note that the subjects, the camera, the lighting, and the viewing angles were all different from the training datasets. The result of the method on a painting further demonstrates the generalization ability.

bounding box and the input features. Qualitative testing showed that with particular facial alignment method used in the work the temporal stability of the method is satisfactory for use in practice.

Chapter 4

Image re-synthesis using deep warping architecture

In this chapter, I proceed to the discussion of the second system (*deep warp*). As in the forest-based approach, the bulk of image re-synthesis is accomplished via warping the input image (Figure 4.1). The task of the network is therefore the prediction of the warping field. This field is predicted in two stages in a coarse-to-fine manner, where the decisions at the fine scale are being informed by the result of the coarse stage. Beyond coarse-to-fine warping, the photorealism of the result is improved by performing pixel-wise correction of the brightness where the amount of correction is again predicted by the network (Figure 4.4). All operations outlined above are implemented in a single feed-forward architecture and are trained jointly end-to-end.

Unlike the warping flow forest system, the deep warp system is trained on pairs of images corresponding to eye appearance before and after the redirection by different angles. The redirection angle serves as an additional input parameter that is provided both during training and at test time. A significant amount of the work, presented in this chapter, was done by my collaborator Yaroslav Ganin. The personal contributions of the thesis author are the experimental setup and the data preparation for the Deep Warp approaches, the comparison of the Deep Warp approach and the forest-based approach, the setup and the implementation of the user study.

4.1 Overview of Deep Learning

Let the data be given by a set $(\mathbf{x}_i, f(\mathbf{x}_i)$. Typically, $\mathbf{x}$ is high-dimensional and $f(\mathbf{x})$ is in $\{0, 1\}$ or $\mathbb{R}$. The goal is to learn how to make accurate predictions in new points, i.e.

f^* – an approximation to f, which is close to f in the given data points.

Following the parametric statistics approach, in deep learning the approximation is from a family if functions $f(\mathbf{x};\theta)$, where θ is a high-dimensional parameter. The goal now is to find θ^* such that $f(\mathbf{x};\theta^*)$ is close to f. The neural network is a composition of functions:

$$f(\mathbf{x},\theta) = f^{(d)}(\cdot,\theta) \circ \cdots \circ f^{(1)}(\cdot,\theta). \tag{4.1}$$

This functions are called layers of the network. Most of them are high-dimensional and depends only of the some subset of the parameters. Components of the vector-valued function $f^{(i)}$ are $h_1^{(i)}, \ldots, h_{n_i}^{(i)}$.

Typically, the layers are rather simple functions, "close" to linear. However, the composition of linear functions is also a linear function. A common design motivated by a neuroscience is a linear function with some non-linear activation:

$$h^{(i)} = g(\mathbf{W}^{(i)^T}\mathbf{x} + \mathbf{b}^{(i)}). \tag{4.2}$$

Here g is the coordinate-wise application of some non-linear function one-dimensional function. Typical choice of activations are:

- RELU (rectified linear units) $g(z) = \max(0, z)$;
- sigmoid function $g(z) = \frac{1}{1+e^{-z}}$;
- hyperbolic tangent $g(z) = \tanh(z)$.

The exception is the top layer, which activation unit often has some statistical interpretation. This choice is tightly coupled with the choice of the cost function, i.e. the criteria used to optimize for parameters. The most typical approach is to train using maximum likelihood, minimizing

$$J(\theta) = \mathbb{E} \log p_{model}(y|\mathbf{x}), \tag{4.3}$$

where the expectation is taken over the data. Thus, the linear function in the top linear (constant activation unit) corresponds to a conditional Gaussian distribution. Choice of a sigmoid function implies thinking of the output as a probability of a Bernoulli trial. The more general softmax output unit

$$softmax(z)_i = \frac{e^{z_i}}{\sum j e^{z_j}}$$

implies Multinoulli Output Distribution.

Peculiar properties of the optimization problem (4.3) are high dimensionality, large data set, non-convex cost function and possible overfitting. A common approach to optimization is a gradient descent. In the particular case of the composition-like structure of a neural network (4.1), an efficient way to do gradient descent is a back-propagation [98], a method that involves clever bookkeeping of partial derivatives by dynamic programming. Because of the large dataset to overcome redundant computations another standard approach is stochastic gradient descent (in particular, mini-batch gradient descent), performing updates only for some small subsets of data at a time. Useful improvements over a vanilla gradient exploit ideas of the second-order methods: Nesterov accelerated gradient [99], Adaptive Moment Estimation [100].

The strategies designed to reduce the test error, possibly at the expense of increased training error, are called regularization. Instead of optimizing $J(\theta)$ (4.3), we optimize over

$$\tilde{J}(\theta) = J(\theta) + \Omega(\theta).$$

Ω often introduces a penalty for complicated or large parameters. For example, L_2 penalty implies a weight decay technique in a gradient descent.

One key type of neural networks are convolutional networks. They can be applied to a data which have a special grid-like topology, for example to images, making them extremely useful in computer vision [38, 101]. Convolutional networks are the neural networks, where at least in one of the layers convolution is used instead of a general matrix multiplication. In comparison with fully connected layer (4.2), convolutional layer has much less parameters, because parameters of convolution depend only on the layer, not on the particular place in the feature map. This both reduces overfitting and makes network to learn useful local features. A common additional ingredient in convolutional networks is pooling, in which, after convolving we replace the result with the average or maximum in a neighborhood. It provides independence of small shifts in the input.

In this work I use batch normalization layer [102]. It is a reparametrization, which is targeted at reducing the problem of coordinating updates across many layers. At training time, each mini-batch of activations of the layer is normalized subtracting mean and dividing on standard deviation. Importantly, the back-propagation is done through these normalizing operations at training time. This means that the gradient descent will not simply increase the standard deviation or mean. At test time, running average mean and deviation from the training time are used.

The work [82] suggests an attention mechanism based on Spatial Transformer Networks. In their method, special part of the network produces the grid, at which input image

or feature maps are further sampled. In the case of pixel-wise replacement approach suggested at this work (Section 2.1), the warping field (2.4) is the grid with the same functions. The work [82] shows how this operation could be implemented in a differentiable layer. An operation with some interpolation kernel k with parameters Φ is written as

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y) \; \forall i \in [1 \dots H'W'] \; \forall c \in [1 \dots C],$$

where (x_i^s, y_i^s) are coordinates of the pixel number i in the sampling grid, U and V are input and output maps, c is a channel number. Interest for this work is a special case of a bilinear kernel (2.6), which reduces the equation above reduces to

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|).$$

Although the summation over the whole grid is written, for each particular pixel i only 4 members are nonzero – those that correspond to neighbor integer points of the point (x_i^s, y_i^s). Therefore, the equation reduces to the simple form of bilinear interpolation (see (2.6), Figure 2.1). However, the equation in the form of a complicated sum is useful for calculating partial derivatives, which are necessary for backpropagation through the sampling mechanism:

$$\frac{\partial V_i^c}{\partial U_{nm}^c} = \sum_n^H \sum_m^W \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|),$$

$$\frac{\partial V_i^c}{\partial x_i^s} = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |y_i^s - n|) \begin{cases} 0, & \text{if } |x_i^s - m| \geq 1; \\ 1, & \text{if } m \geq x_i^s; \\ -1, & \text{if } m < x_i^s. \end{cases}$$

and similarly for $\frac{\partial V_i^c}{\partial y_i^s}$.

4.2 Coarse-to-fine warping

The warping modules takes as an input the image, the position of the feature points, and the redirection angle. All inputs are expressed as maps as discussed below, and the architecture of the warping modules is thus "fully-convolutional", including several convolutional layers interleaved with Batch Normalization layers [102] and ReLU nonlinearities (the actual configuration is shown in Figure 4.2(c)). To preserve the resolution of

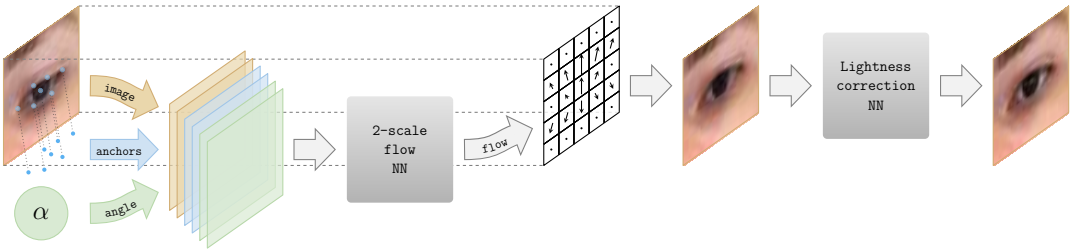
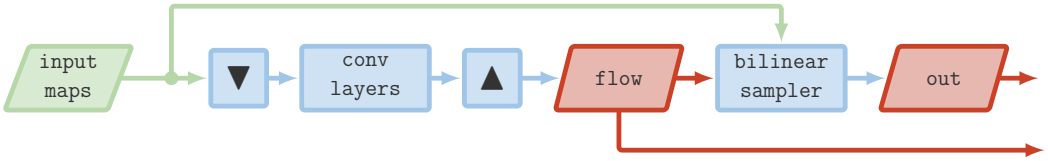
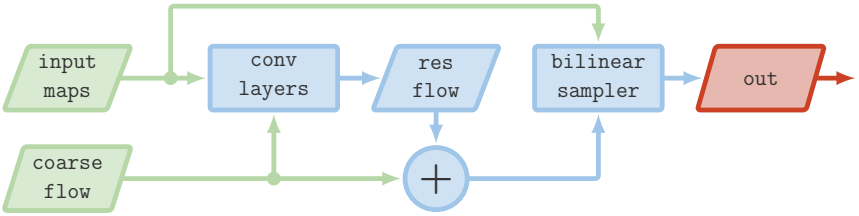


FIGURE 4.1: The deep warp system takes an input eye region, feature points (`anchors`) as well as a correction `angle` α and sends them to the multiscale neural network (see Section 4.2) predicting a `flow` field. The flow field is then applied to the input image to produce an image of a redirected eye. Finally, the output is enhanced by processing with the lightness correction neural network (see Section 4.4).



(a) 0.5×-scale processing module



(b) 1×-scale processing module

(c) Convolutional layers

FIGURE 4.2: The architecture of the two warping modules: (`process` 0.5×`-scale` 4.2(a) and `process` 1×`-scale` 4.2(b)) predicting and applying pixel-flow to the input image; 4.2(c) represents a fully convolutional sequence of layers inside warping modules.

the input image, the 'same'-mode convolutions (with zero padding) are used, all strides are set to one, and max-pooling is avoided.

The resulting flow is obtained using coarse-to-fine two-stage warping. Firstly, the first part (stage) of the network estimates the coarse flow at half resolution. Then, the second stage of the network performs additive rectification at full scale, using the upsampled coarse flow as well as the feature maps computed by the first-stage. The details are provided below.

Coarse warping. The last convolutional layer of the first (half-scale) warping module (Figure 4.2(a)) produces a pixel-flow field (a two-channel map), which is then upsampled $\mathcal{F}_{\text{coarse}}(I, \alpha)$ and applied to warp the input image by means of a bilinear sampler $\mathbf{S}$ [82]

that finds the *coarse estimate*:

$$O_{\text{coarse}} = \mathbf{S}\left(I, \mathcal{F}_{\text{coarse}}(I, \alpha)\right) .$$

Here, the sampling procedure S samples the pixels of O_{coarse} at pixels determined by the flow field (the procedure described in Section 2.1, (2.3), (2.5)).

Fine warping. In the fine warping module (Figure 4.2(b)), the rough image estimate O_{coarse} and the upsampled low-resolution flow $\mathcal{F}_{\text{coarse}}(I, \alpha)$ are concatenated with the input data (the image, the angle encoding, and the feature point encoding) at the original scale and sent to the $1\times$-scale network which predicts another two-channel flow $\mathcal{F}_{\text{res}}$ that amends the half-scale pixel-flow (additively [103]):

$$\mathcal{F}(I, \alpha) = \mathcal{F}_{\text{coarse}}(I, \alpha) + \mathcal{F}_{\text{res}}(I, \alpha, O_{\text{coarse}}, \mathcal{F}_{\text{coarse}}(I, \alpha)) ,$$

the amended flow is used to obtain the final output (again, via bilinear sampler):

$$O = \mathbf{S}\left(I, \mathcal{F}(I, \alpha)\right) .$$

The purpose of coarse-to-fine processing is two-fold. The half-scale (coarse) module effectively increases the receptive field of the model resulting in a flow that moves larger structures in a more coherent way. Secondly, the coarse module gives a rough estimate of how a redirected eye would look like. This is useful for locating problematic regions which can only be fixed at a finer scale.

4.3 Input embedding

Alongside the raw input image, the warping modules also receive the information about the desired redirection angle and feature points also encoded as image-sized feature maps.

Embedding the angle. Similarly to [43], the correction angle is treated as an attribute and is embedded into a higher dimensional space using a multilayer perceptron $\mathbf{F}_{\text{angle}}(\alpha)$ with ReLU nonlinearities. The precise architecture is `FC(16)` $\rightarrow$ `ReLU` $\rightarrow$ `FC(16)` $\rightarrow$ `ReLU`. Unlike [43], separate features are not outputted for each spatial location but rather opt for a single position-independent 16-dimensional vector. The vector is then expressed as 16 constant maps that are concatenated into the input map stack. During learning, the embedding of the angle parameter is also updated by backpropagation.

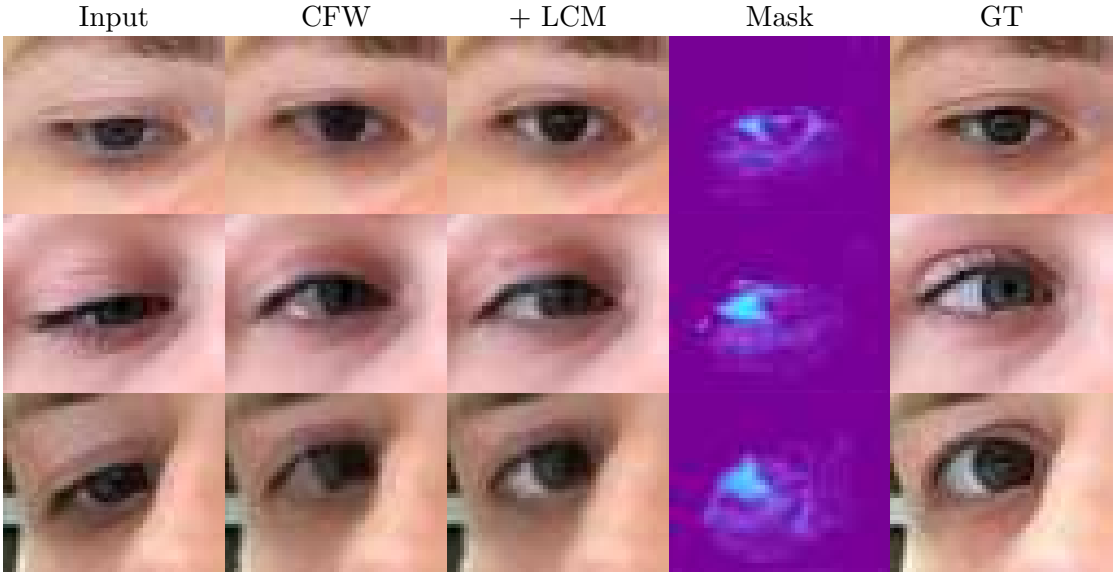


FIGURE 4.3: Visualization of three challenging redirection cases where the **Lightness Correction Module** helps considerably compared to the system based solely on coarse-to-fine warping (CFW), which is having difficulties with expanding the area to the left of the iris. The 'Mask' column shows the soft mask corresponding to parts where lightness is increased. Lightness correction fixes problems with dis-occluded eye-white, and also emphasizes the specular highlight increasing the perceived realism of the result.

Embedding the feature points. Although in theory a convolutional neural network of an appropriate architecture should be able to extract necessary features from the raw input pixels, it is beneficial to further augment 3 color channels with additional 14 feature maps containing information about the eye anchor points.

In order to get the anchor maps, for each previously obtained feature point located at $\mathbf{l}_i = (l_i^x, l_i^y)$, a pair of maps is computed:

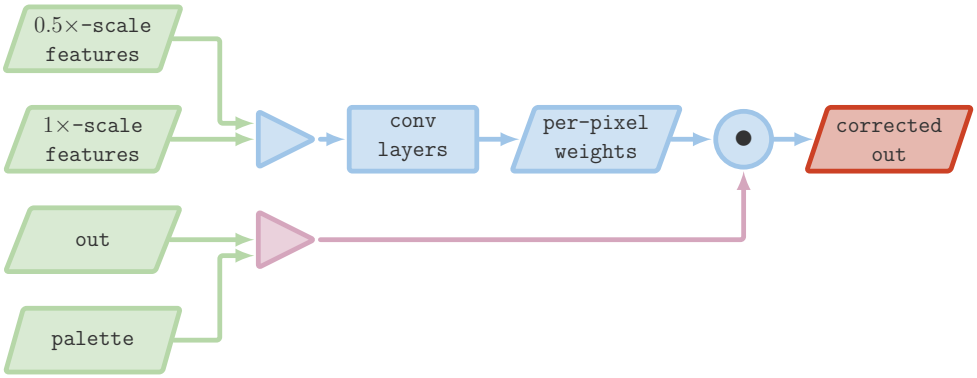
$$\begin{aligned} \Delta_x^i[x,y] &= x - l_i^x, \\ \Delta_y^i[x,y] &= y - l_i^y, \end{aligned} \quad \forall (x,y) \in \{0,\ldots,W\} \times \{0,\ldots,H\},$$

where W, H are width and height of the input image respectively. The embedding give the network "local" access to similar features as used by decision trees.

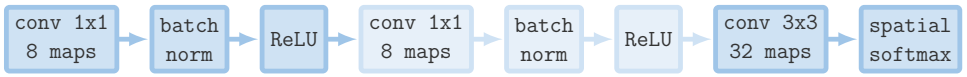
Ultimately, the input map stack consists of 33 maps (RGB + 16 angle embedding maps + 14 feature point embedding maps).

4.4 Lightness correction module

While the bulk of appearance changes associated with gaze redirection can be modeled using warping, some subtle but important transformations are more photometric than



(a) Architecture.



(b) Convolutional layers.

FIGURE 4.4: 4.4(a) – The architecture of the Lightness Correction Module. The output of the lightness correction module is a weighted sum of the image created by the warping modules and the palette (which in this paper is taken to be a single white colour). The mixing weights predicted by the network are passed through the softmax activation and therefore sum to one at each pixel. The module takes the features computed by the coarse and the fine warping modules (from Figure 4.4(b)) as input.

geometric and require a more general transformation. In addition, the warping approach can struggle to fill in dis-occluded areas in some cases (Figure 4.3).

To increase the generality of the transformation that can be handled by the deep warp architecture, the final lightness adjustment module is added (Figure 4.4(a)). The module takes as input the features computed within the coarse warping and the fine warping modules (specifically, the activations of the third convolutional layer), as well as the overall image resulting from the warping. The output of the module is a single map M of the same size as the output image that is used to modify the brightness of the output O using a simple element-wise transform:

$$O_{\text{final}}(x, y, c) = O(x, y, c) \cdot (1 - M(x, y)) + M(x, y), \tag{4.4}$$

assuming that the brightness in each channel is encoded between zero and one. The resulting pixel colors can thus be regarded as blends between the colors of the warped pixels and the white color. The actual architecture for the lightness correction module in experiments is shown in Figure 4.4(b).

4.5 Training procedure

A regular ℓ_2-distance between the synthesized output O_{output} and the ground-truth O_{gt} is used as the objective function. The model was trained end-to-end on 128-sized batches using Adam optimizer [100]. Biasing the selection process for more difficult and unusual head poses and bigger redirection angles improved the results. For this reason, the following sampling scheme aimed at reducing the dataset imbalance is used:

- Split all possible correction angles (that is, the range between $-30°$ and $30°$) into 15 bins.
- A set of samples falling into a bin is further divided into "easy" and "hard" subsets depending on the input's *tilt* angle (an angle between the segment connecting two most distant eye feature points and the horizontal baseline). A sample is considered to be "hard" if its tilt is $\geqslant 8°$. his subdivision helps to identify training pairs corresponding to the rare head poses. A training batch is formed by picking 4 correction angle bins uniformly at random and sampling 24 "easy" and 8 "hard" examples for each of the chosen bins.

4.6 Experiments

In this section computational experiments, illustrating the suggested Deep Warp approach, are provided. Both quantitative and qualitative results are given. More results could be found in comparisons with other methods in the following chapters (Section 5.3 and Section 6.4).

4.6.1 Quantitative evaluation

Experiments are performed on a Skoltech dataset, described in Section 2.3. The initial set of subjects is randomly splitted into a development (26 persons) and a test (7 persons) sets. Several methods were compared using the mean square error (MSE) between the synthesized and the ground-truth images extracted using the procedure described in (2.8).

4.6.1.1 Models.

Six different models are considered:

(1) A system based on weakly-supervised Random Forests (*RF*) described in Chapter 3.

(2) A single-scale (*SS* (15° only)) version of DeepWarp method with a single warping module operating on the original image scale that is trained for 15° redirection only. Single-scale here denotes, that a model does not exploit the coarse-to-fine idea, but only produces one warping field on the original image scale. Lightness correction module is also not used. This model is similar to the one presented in Figure 4.2(a), but without downsampling and upsampling.

(3) A single-scale (*SS*) version of DeepWarp method with a single warping module operating on the original image scale.

(4) A multiscale (*MS*) network without coarse warping. In this variation, the coarse warping is not amended on a fine scale, but features from both scales, collected independently, are used to predict the warping field. Lightness correction module is also not used.

(5) A coarse-to-fine warping-based system described in Section 4.2 (*CFW*).

(6) A coarse-to-fine warping-based system with a lightness correction module (*CFW + LCM*).

The latter four models are trained for the task of vertical gaze redirection in the range. Such models are called *unified* (as opposed to single angle correction systems).

4.6.1.2 15° correction.

In order to have the common ground with the forest-based system, this comparison is restricted to the case of 15° gaze correction. Following the same approach as in Section 3.3, a graph of sorted normalized errors (Figure 4.5) is presented, where all errors are divided by the MSE obtained by an input image and then the errors on the test set are sorted for each model.

It can be seen that the unified multiscale models are, in general, comparable or superior to the RF-based approach in Chapter 3. Interestingly, the lightness adjustment extension (Section 4.4) is able to show quite significant improvements for the samples with low MSE. Those are are mostly cases similar to shown in Figure 4.3. It is also worth noting that the single-scale model trained for this specific correction angle consistently outperforms Chapter 3, demonstrating the power of the proposed architecture. However, one should note that results of the methods can be improved using additional registration procedure, one example of which is described in Section 4.6.3.

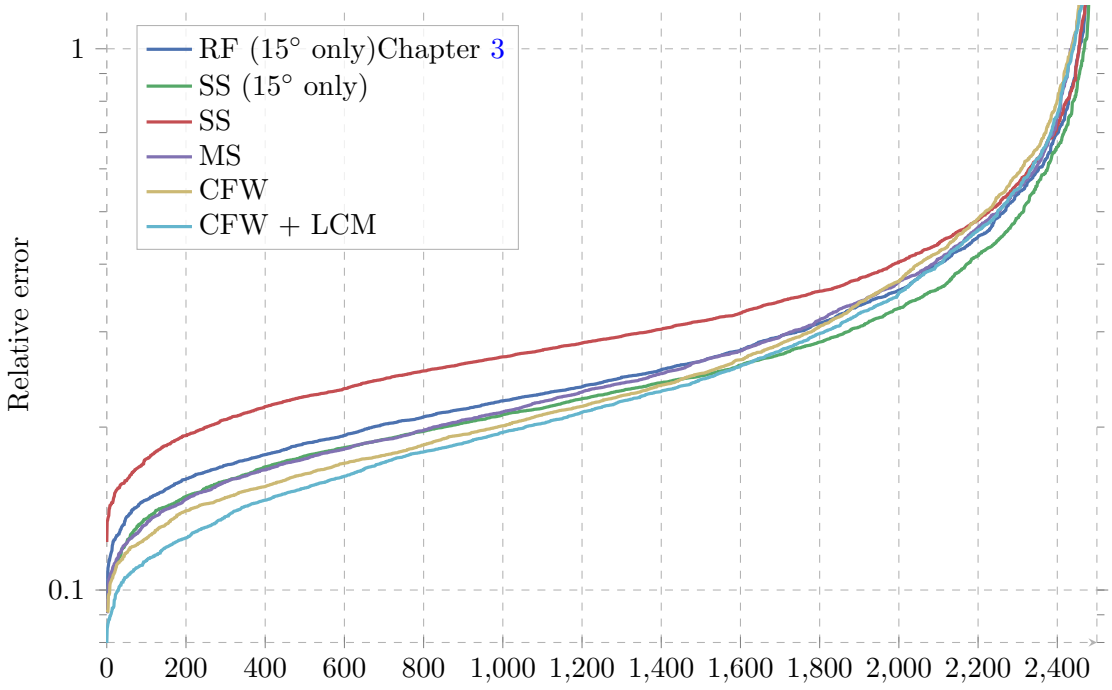


FIGURE 4.5: Ordered errors for 15° redirection. Multiscale models (MS, CFW, CFW + LCM) show results that are comparable or superior the Random Forests (RF) Chapter 3.

4.6.1.3 Arbitrary vertical redirection.

In Figure 4.6 different variants of unified networks are compared and the error distribution over different redirection angles is plotted. The neural network models in this comparison are trained for the task of vertical gaze redirection in the range from $-30°$ to $30°$. For small angles, all the methods demonstrate roughly the same performance, but as the amount of correction is increased, the task becomes much harder (which is reflected by the growing error) revealing the difference between the models. Again, the best results are achieved by the palette model, which is followed by the multiscale networks making use of coarse warping.

4.6.2 Perceptual quality

The results of redirection on 15 degrees upwards are demonstrated in (Figure 4.7). CFW-based systems produces the results visually closer to ground truth, than RF. The effect of the lightness correctness is pronounced: on the input image with the lack of white Random Forest and CFW fail to get output with sufficient eye-white and copy-paste

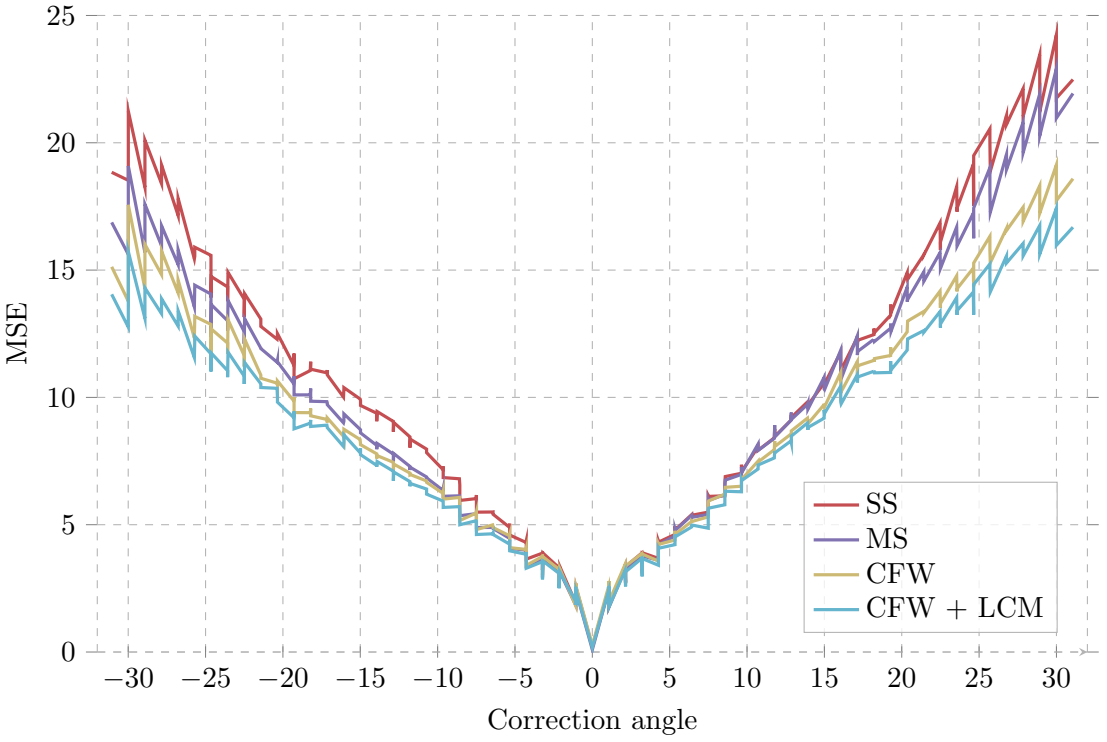


FIGURE 4.6: Distribution of errors over different correction angles.

red pixels instead, whereas CFW+LCM achieve good correspondence with the ground-truth. However, the downside effect of LCM could be blurring/lower contrast because of the multiplication procedure (4.4).

4.6.2.1 User study

To confirm the improvement corresponding to different aspects of the proposed models, which may not be adequately reflected by an ℓ_2-measure, I perform an informal user study enrolling 16 subjects unrelated to computer vision and comparing four methods (RF, SS, CFW, CFW+LCM). Each user was shown 160 quadruplets of images, and in each quadruplet one of the images was obtained by re-synthesis with one of the methods, while the remaining three were unprocessed real images of eyes. 40 randomly sampled results from each of the compared methods were thus embedded. When a quadruplet was shown, the task of the subject was to click on the artificial (re-synthesized) image as quickly as possible. For each method, I then recorded the number of correct guesses out of 40 (for an ideal method the expected number would be 10, and for a very poor one it would be 40). I also recorded the time that the subject took to decide on each quadruplet (better method would take a longer time for spotting). Table 4.1 shows results of the experiment. Notably, here the gap between methods is much wider then it

FIGURE 4.7: Sample results on a hold-out. The full version of DeepWarp model (CFW+LCM) outperforms other methods.

	Random Forest	Single Scale	CFW	CFW+LCM
Correctly guessed (out of 40)				
Mean	36.1	33.8	28.8	**25.3**
Median	37	35	29	**25**
Max	40	39	38	**34**
Min	26	22	20	**16**
Correctly guessed within 2 seconds (out of 40)				
Mean	26.4	21.1	11.7	**8.0**
Median	28.5	20.5	10	**8**
Max	35	33	23	**17**
Min	13	11	3	**0**
Correctly guessed within 1 second (out of 40)				
Mean	8.1	4.4	1.6	**1.1**
Median	6	3	**1**	**1**
Max	20	15	7	**5**
Min	**0**	**0**	**0**	**0**
Mean time to make a guess				
Mean time, sec	1.89	2.30	3.60	**3.96**

TABLE 4.1: **User assessment for the photorealism of the results for the four methods.** During the session, each of the 16 test subjects observed 40 instances of results of each method embedded within 3 real eye images. The participants were asked to click on the resynthesized image in as little time as they could. The first three parts of the table specify the number of correct guesses (the smaller the better). The last line indicates the mean time needed to make a guess (the larger the better). The full system (coarse-to-fine warping and lightness correction) dominated the performance.

might seem from the MSE-based comparisons, with CFW+LCM method outperforming others very considerably, especially when taking into account the timings.

4.6.2.2 Continuous gaze redirection.

The deep warp model is capable of a redirection by a 2D family of angles, as stated, as the input angle is embedded in the architecture Figure 4.1. In Figure 4.8 and Figure 4.9, qualitative results of CFW+LCM for vertical and horizontal redirection are provided. Some examples showing the limitations of the method are given. The limitations are concerned with cases with severe dis-occlusions, where large areas have to be filled by the network.

4.6.3 Incorporating registration.

Results can be further perceptually improved if the objective is slightly modified to take into account misalignment between inputs and ground-truth images. To that end, enlarge the bounding-box $\mathcal{B}$ which was used to extract the output image of a training

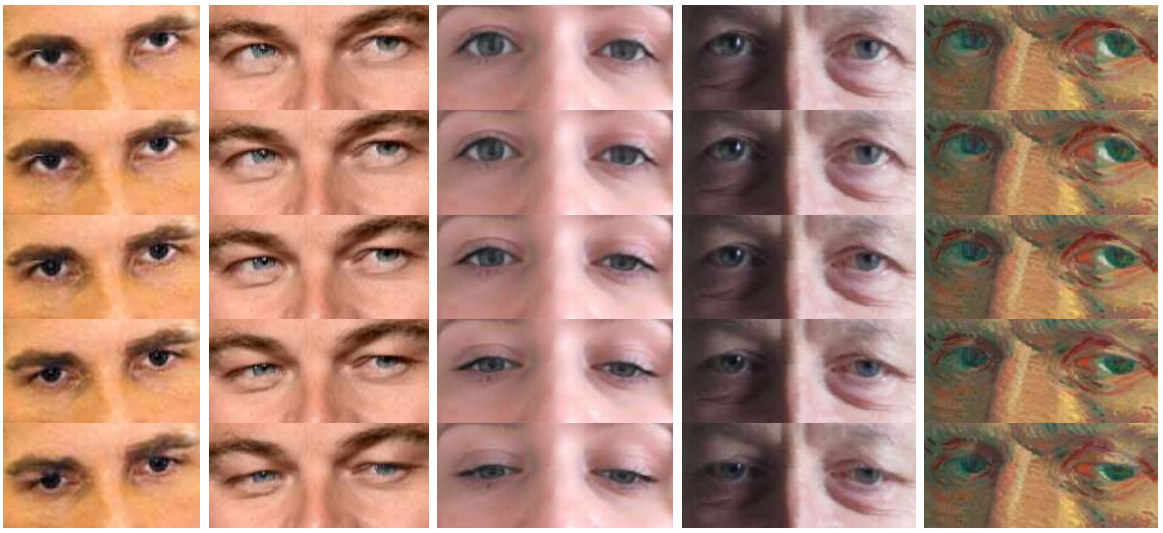


FIGURE 4.8: Gaze redirection with deep warp model trained for vertical gaze redirection. The model takes an input image (middle row) and the desired redirection angle (here varying between -15 and +15 degrees) and re-synthesize the new image with the new gaze direction. Note the preservation of fine details including specular highlights in the resynthesized images.

pair by $k = 3$ pixels in all the directions. Given that now O_{gt} has the size of $(H + 2k) \times (W + 2k)$, the new objective is defined as:

$$\mathcal{L}(O_{\text{output}}, O_{\text{gt}}) = \min_{i,j} \text{dist}\left(O_{\text{output}}, O_{\text{gt}}[i : i + H, j : j + W]\right) ,$$

where dist$(\cdot)$ can be either ℓ_2 or ℓ_1-distance (the latter giving slightly sharper results), and $O_{\text{gt}}[i : i + H, j : j + W]$ corresponds to a $H \times W$ crop of O_{gt} with top left corner at the position (i, j). This procedure is an alternative to the offline registration of input/ground-truth pairs described in Section 3.2.4. In case of redirection on an arbitrary angle that procedure is computationally intractable for a large database, because number of training pairs from each training sequence (which should be registered) depends quadratically on the sequence size. This dependence is linear in case when only training pairs with some fixed angular difference are taken, i.e. 15°. The suggested approach is computationally cheap and increases robustness of the training procedure against small misalignments in a training set.

FIGURE 4.9: Horizontal redirection with a model trained for both vertical and horizontal gaze redirection. For the first six rows the angle varies from $-15°$ to $15°$ relative to the central (input) image. The last two rows push the redirection to extreme angles (up to $45°$) breaking the model down.

Chapter 5

Regression random forest using neural network supervision

The quality of the results of suggested in Chapter 4 deep warping system compares favorably with the results of the warping flow forest-based system, presented in Chapter 3. One more drawback of this system is its big memory footprint. This is because storing the distributions of the compatibility score (3.12) in the leaves requires the amount of memory proportional to the patch size. In my implementation, training warping flow forests with several trees and 9-by-9 patches leads to the size of the resulting model up to 200Mb.

The better quality of the deep warp results come at the cost of the much higher computation time (few frames per second on GPU). At this chapter I suggest the system based on *neural network-supervised forests*. It aims at combining the speed of the forest-based system with the quality of the deep warp system. This is achieved by training regression forests to emulate the predictions of the deep warp system at each pixel. Assuming that during training, the prediction of deep warp are treated as pseudo ground truth, the regression forests can be trained in a traditional fully-supervised manner discussed below.

5.1 Related work on teacher-student architectures

Training a regression forests to emulate the predictions of the neural network relates this work to several recent papers with similar teacher-student architectures. The idea to use output of very precise but large and slow model as a supervision for a faster architecture goes back to at least [78]. They have shown that the model ensemble could

be compressed into a single much faster model. A different kind of training is often needed to transfer the knowledge from the teacher model to a small model. The idea is to pass unlabeled data through the large, accurate model to collect the scores produced by that model. A fast and compact model does not overfit approximates well the target function, provided enough pseudo data by high performing model ensemble. Work [78] suggests several ways to generate a new pseudo data, estimating the joint distribution of attributes in the original training set for large model and drawing points from this generative model trained on the original training set.

An alternative way is to train the smaller model only on the original training points, but train it to copy other features of the model, such as its posterior distribution over the full set of classes [79]. They suggested to train a "student" network, from the softened output of an ensemble of wider networks ("teacher" networks), allowing the student network to capture not only the information provided by the true labels, but also the finer structure learned by the teacher network. In such a way an ensemble of deep networks is compressed into a student network of similar depth. In particular, they use a softened version of softmax teacher output

$$q_i = \frac{e^{\frac{z_i}{T}}}{\sum\limits_j e^{\frac{z_j}{T}}} \tag{5.1}$$

with temperature $T > 1$ (the case $T = 1$ corresponds to the usual softmax). They train the student network on a weighted average of two different objective functions. The first objective function is the cross entropy with the soft targets (5.1). This cross entropy is also modified using the same high temperature in the softmax of the distilled model as was used for generating the soft targets from the teacher model. The second objective function is the cross entropy with the correct labels. This cross entropy is computed using the normal softmax at a temperature of 1.

This idea was further developed in [77] that uses activations of several hidden layers of the teacher network as a guidance for the student network. In this approach the teacher network has low enough depth and great enough width to be easy to train. The student network is much deeper and thinner. Activations of several hidden layers, provided for optimization, simplify the optimization problem. Additional layer is added to the student network, which goal is to regress the middle layer of the teacher network from the middle layer of the student network. The corresponding error gradient in this architecture influence only the lower layers of the student network. Thus they have two objectives: to help the higher layers of the student network to predict the label, and to model the intermediate layer of the teacher network. It is shown in the experiments

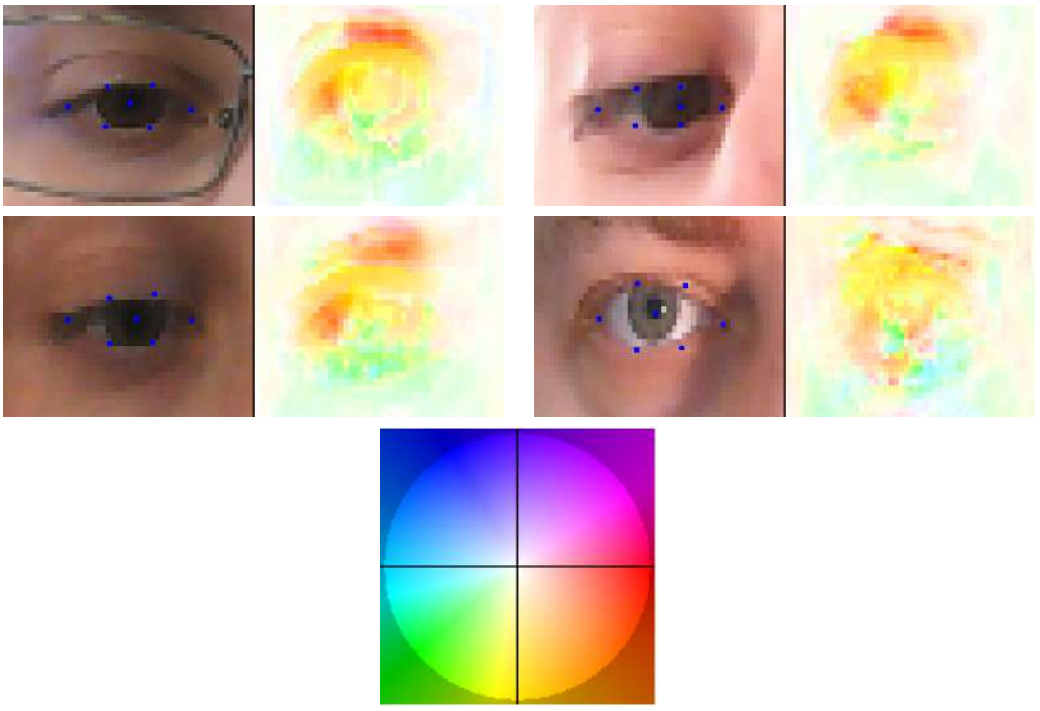


FIGURE 5.1: The output flow of warping modules (Section 4.2) on random samples from a training set. The coarse-to-fine model without lightness correction was trained on a task of 15° redirection upwards. This data is used to train a neural network-supervised regression random forest. The bottom figure is a color pattern, explaining how the direction of the flow vector is encoded with the color of the pixel. The more intense the pixel is, the longer the flow vector in this pixel is.

that such hints on middle layers improves the results of student network (which learns very poorly without hints) both on train and test set.

5.2 Learning

To learn the system, I fix the desired redirection angle and use the sum of the coarse and the fine flow predicted by a deep warp system (the *teacher*) for the given angle as ground-truth data, thus directly predicting the warping field (2.4). The examples of such training data are shown in Figure 5.1.

The input data to the NN-supervised regression forests is the same as to the weakly supervised warping flow forests, i.e. an image of the eye and its landmark locations. The regression forests apply the same appearance and location tests ((3.10) and (3.11)) as the warping flow forests. A set of training samples is $\mathbf{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K\}$, where each sample is a tuple $\mathcal{S} = \{(x, y), I, \{\mathbf{l}_i\}\}, (dx, dy)\}$ with dx and dy being the prediction of the deep warp system. The quality of the split of $\mathbf{S}$ into two subsets $\mathbf{S}_1$ and $\mathbf{S}_2$ is the

common objective for regression problems ((3.4), (3.7)):

$$I(\mathbf{S}_1, \mathbf{S}_2) = H(\mathbf{S}) - \left(\frac{|\mathbf{S}_1|}{|\mathbf{S}|} H(\mathbf{S}_1) + \frac{|\mathbf{S}_2|}{|\mathbf{S}|} H(\mathbf{S}_2) \right), \tag{5.2}$$

where compatibility scores are

$$H(\mathbf{S}_i) = H_x(\mathbf{S}_i) + H_y(\mathbf{S}_i) = \sum_{s \in S_i} (dx_s - \overline{dx}_i)^2 + \sum_{s \in S_i} (dy_s - \overline{dy}_i)^2,$$

with $\overline{dx}_i$ and $\overline{dy}_i$ being the means of dx and dy flow in the subset $\mathbf{S}_i$.

At test time, a pixel is passed down the tree, and the mean 2D flow vector (across the training examples, which fell to the corresponding leaf) is picked. The flows coming from different trees in a forest are averaged to get the final result. As the range of flow is limited, only two bytes are sufficient to store a flow vector in each leaf, which is much smaller compared to weakly supervised forest. The reduction in memory is thus $(w^2 - 2)$ bytes for each leaf, where w is the size of the width of the patch in the weakly-supervised warping flow forest system (Figure 3.2). Moreover, as the supervised task is easier to learn, the depth and the number of leaves can be made lower, resulting in typical memory demand for trained forests of only three megabytes.

The prediction of the lightness correction module Section 4.4 can be incorporated using the second forest, which is applied to the output images after applying the warping predicted by the first NN-supervised forest. The train set for the second forest is the output of the lightness correction module of the neural network and images $\{\hat{I}^j\}$, which are obtained from original set $\{I^j\}$ by applying the first forest and warping procedure (2.2). The splitting criterion is the same as (5.2), except that output in this case is a one-dimensional vector.

So, the computational speed of weakly supervised in nn-supervised tree of the same depth would be roughly the same. However, while weakly supervised forest learns to predict the flow, having only the output image as a supervision, the nn-supervised uses target flow directly as an output of neural network warping layers. If the student forest would be able to approach the precision of the teacher neural network model, then it will take benefits of both approaches: real-time computational speed from a random forest approach and high quality of the results of a teacher network, both in terms of quantitative ℓ_2 measure and a perceptual quality. In Section 5.3 I provide the comparison showing that the ability of the regression forest to learn from a teacher is high enough to approach its quality and to outperform a quality of a weakly supervised forest.

5.3 Experiments

In this section I compare the performance of neural network-supervised forests with weakly-supervised forest form Chapter 3 and deep warping approach from Chapter 4.

5.3.1 Evaluation using Mean Squared Error

I evaluate the methods on a Skoltech dataset. I randomly split the initial set of subjects into the training and the testing sets. I sample image pairs applying the same preprocessing steps as when preparing data for learning (Section 2.3).

15° **correction.** First I provide a comparison of MSE errors in case of 15° vertical gaze redirection. For this comparison the following models are considered:

1. A system based on weakly supervised warping flow random forests (*EFF*), described in Chapter 3. The score of this model is what random forest could achieve without supervision from neural network.
2. A coarse-to-fine warping-based system described in Chapter 4 (*CFW*) without lightness correction trained for 15° vertical gaze redirection. The results of this model are the gold standard for the student forest.
3. A neural network supervised random forest (*NNSF*), described in this chapter in Section 5.2, which predicts the output flow of a coarse-to-fine warping-based system without lightness correction. The teacher for the method is *CFW* (the resulting flow on train set).
4. A deep warp coarse-to-fine warping-based system with a lightness correction module trained for 15° vertical gaze redirection (*CFW + LCM*) (Section 4.4) – to test the effect of lightness correction model on the score.
5. A simple baseline – image independent flow field (Section 3.2.1), where the flow is based solely on the relative position of the pixel (*IIF*). The score of this model is in some sense the basic score of warping approach – what we can get without any grouping of similar pixels or representation learning.

In Figure 5.2 I present a graph of sorted normalized errors, where all errors are divided by the MSE obtained by an input image taken as an output. For each method, the errors on the test set are sorted. It can be seen, that NN-supervised forest performs comparably to the warping flow forest. What is more important, this improvement is quite visible in terms of noise and artifacts (Figure 5.4). However, there is still a gap

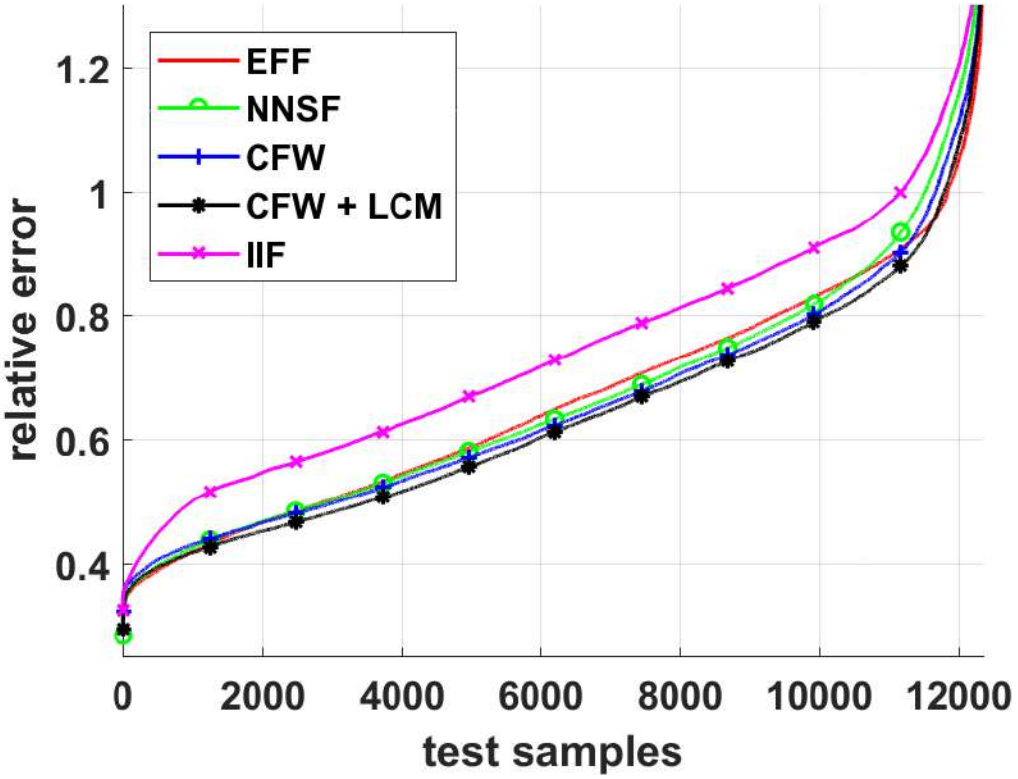


FIGURE 5.2: Ordered errors for 15° vertical gaze redirection (see text for more discussion of the error metric). The best performance is shown by the full coarse-to-fine architecture with the lightness correction module.

between the NN-supervised forest and its teacher, the multiscale model without the lightness correction module. The lightness adjustment extension (Section 4.4) is able to show quite significant improvements. Those are mostly cases similar to shown in Figure 4.3. Unified multiscale models trained to handle different angles (not included in this comparison) are, in general, comparable with the one specialized for 15° redirection. It is also worth noting that even a single-scale model trained for this specific correction angle consistently outperforms warping flow forest, demonstrating the superiority of deep learning.

5.3.2 Was the gaze actually redirected?

The low error between output and ground truth does not fully answer the question, whether the gaze difference between input image and output equals requested angle. One could imagine the example (possibly degenerate), when the ℓ_2 error is low and the results are realistic, but the gaze was not redirected or was redirected by a smaller angle than the requested. Thus, I provide additional assessment on evaluating the redirection angle on the test set.

Firstly, the tool for evaluating the redirection angle should be developed. I use the evaluation model E, trained to determine the angular difference between two input images of eyes. In order to increase the accuracy of this model, I consider only vertical redirections. The model is parametrized as deep neural network. The inputs to network are two images of the eye, and the output is the angular gaze difference. The network was trained with MSE loss on pairs of images of same eyes with different vertical gaze directions, picked up from the training part of the dataset. For each training example, the sequence and the horizontal gaze direction in this sequence were picked up randomly. Then, two vertical gaze directions were chosen randomly (the maximum vertical gaze difference in the database is 36°, Section 2.3). The mean RMSE on the training set is 0.8°, and on the validation set is 1.1°.

The architecture of the network E consists of two parts. I denote $\mathrm{conv}(m, k)$ a convolutional layer with m maps and kernel size k, and $\mathrm{FC}(m)$ a fully connected layer with m maps, both precede the RELU activation. The first part is convolutional network with max-pooling, which is applied to both input images with shared weights:

$$\mathrm{conv}(48, 3) \to \mathrm{conv}(48, 3) \to \mathrm{MaxPool} \to \mathrm{conv}(48, 3) \to \mathrm{conv}(48, 3) \to \mathrm{MaxPool}.$$

Then the maps corresponding to two inputs are concatenated, and the second part is applied:

$$\begin{aligned}&\mathrm{conv}(96, 3) \to \mathrm{conv}(96, 3) \to \mathrm{MaxPool} \to \mathrm{conv}(96, 3) \to \mathrm{conv}(96, 3) \to \mathrm{MaxPool}\\ &\to \mathrm{FC}(1000) \to \mathrm{FC}(250) \to \mathrm{FC}(1),\end{aligned}$$

where the last activation is linear.

Using this model, the results of 15° upwards redirection on the testing set were evaluated. For each input image I, the resulting image O_m with gaze redirected 15° upwards was obtained by each method. Then, the actual redirection angle was estimated as $\alpha \approx E(I, O_m)$. The results of this assessment are presented in Figure 5.3. The distribution of the outputs of evaluation model E are plotted for each method. For reference, I also plot the distribution for the ground truth images. The mean of the distribution for the ground truth images is exactly at 15°, but the variance is quite significant. This variance is explained by the inaccuracy in dataset labels and imperfection of the evaluation network. The mean of distributions for all models come quite close to 15°, while the variance does not exceed the variance of the distribution corresponding to the ground truth pairs. Thus, I conclude that all methods performed well in this test, which suggests that the methods should be compared primarily based on the realism of their results. Interestingly, the best performance was shown by the Deep Warp model without lightness correction module, although LCM decreases the ℓ_2 error. The means of forest

based models are slightly less than 15°, while the mean of CFW+LCM model is slightly higher than 15°.

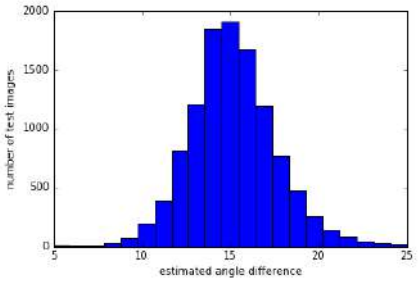
5.3.3 Qualitative evaluation

To qualitatively compare the systems, I show a *random* subset of results of redirection by 15 degrees upwards in Figure 5.4. All methods were trained for 15° vertical redirection. One can observe that the system is able to learn to redirect the gaze of unseen people rather reliably obtaining a close match with the ground truth. Deep warp systems produce the results visually closer to the ground truth, than forest-based systems. The effect of lightness correction is pronounced: on the input image with the invisible sclera in one corner, the system with lightness correction performs clearly better. However, the downside effect of lightness correction could be blurring/lower contrast because of the multiplication procedure (4.4).

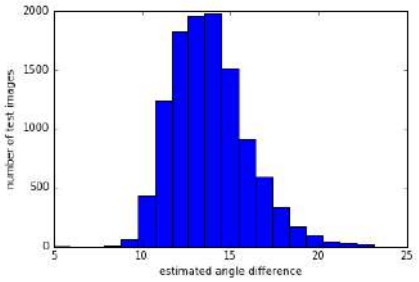
Examples of predicted warping field are also shown in Figure 5.5. To determine the inherent dimensionality of the warping fields manifold, the following experiment was conducted. Warping fields predicted on the validation set were resized to the same size of 80×100 pixels, aligned to vectors, and the PCA model was learned. For the learned model, the dependence of the percentage of explained variance on the number of PCA components is plotted in Figure 5.6. Almost 95% of the variance is described by 100 components out of $80 \times 100 \times 2 = 16000$, and all components after number 1000 have less than 0.001% contribution. It can be viewed as a confirmation, that the predictor had learned the inner structure of a transformation.

Lower resolution images. In Figure 5.7 I provide results of the NNSF-system for lower input resolutions on randomly sampled images from a test set. For that, I downsample images from the original size of 80×100, to 10×12, 20×25, 40×50 correspondingly. The NNSF-system is then applied to the downsampled versions. The effect of gaze redirection is noticeable even for very low resolutions.

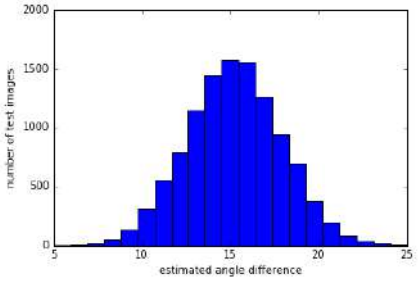
Comparison with the previous work on Gaze Correction. In Figure 5.8 I show the side-by-side comparison with the system [17], which also performs monocular gaze correction for the videoconferencing scenario. The difference in the approaches is clearly visible. Suggested method redirects gaze and confines the changes to the eye region, while keeping the head pose unchanged. On the contrast, the system [17] synthesizes the novel view for the facial part then blending the new face area into the input image. The latter approach results in certain distortion of face proportions. Moreover, [17]



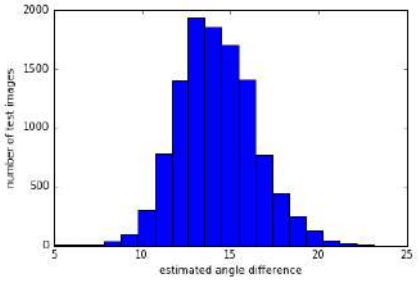
The distribution corresponding to the pairs with 15° difference according to the data collection procedure. Mean = 15.1°, std = 2.6°.



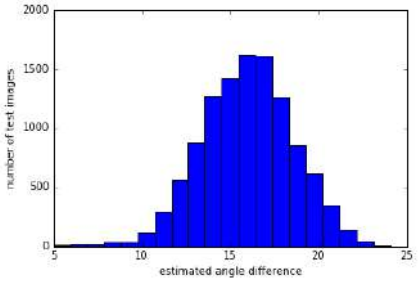
The weakly supervised random forest (Chapter 3). Mean = 13.8°, std = 2.1°.



The coarse-to-fine warping-based system (Chapter 4). Mean = 15.2°, std = 2.7°.



The random forest supervised by a neural network (Chapter 5). Mean = 14.2°, std = 2.2°.



The Deep Warp coarse-to-fine warping-based system with a lightness correction module (Chapter 4). Mean = 16.0°, std = 2.6°.

FIGURE 5.3: The assessment of the redirection angles. Hold-out pairs with gaze difference of 15 degrees were submitted to the network, trained to determine the vertical angular difference. The distribution of the redirection angle, as predicted by the network, is plotted. For the reference, the distribution for the ground truth images is shown in the top row. Means of distributions for all models come quite close to 15°, while the variance does not exceed the variance of the reference distribution. The best results are shown by Deep Warp architecture without lightness correction model.

FIGURE 5.4: Results on a random subset of the hold-out test set. From left to right: (a) Input, (b) Warping flow forests, (c) Neural network supervised forests, (d) Coarse-to-fine warping with the lightness correction module, (e) Coarse-to-fine warping without the lightness correction module, (f) Ground truth. The full variant (CFW + LCM) of deep warp system (e) generally performs the best.

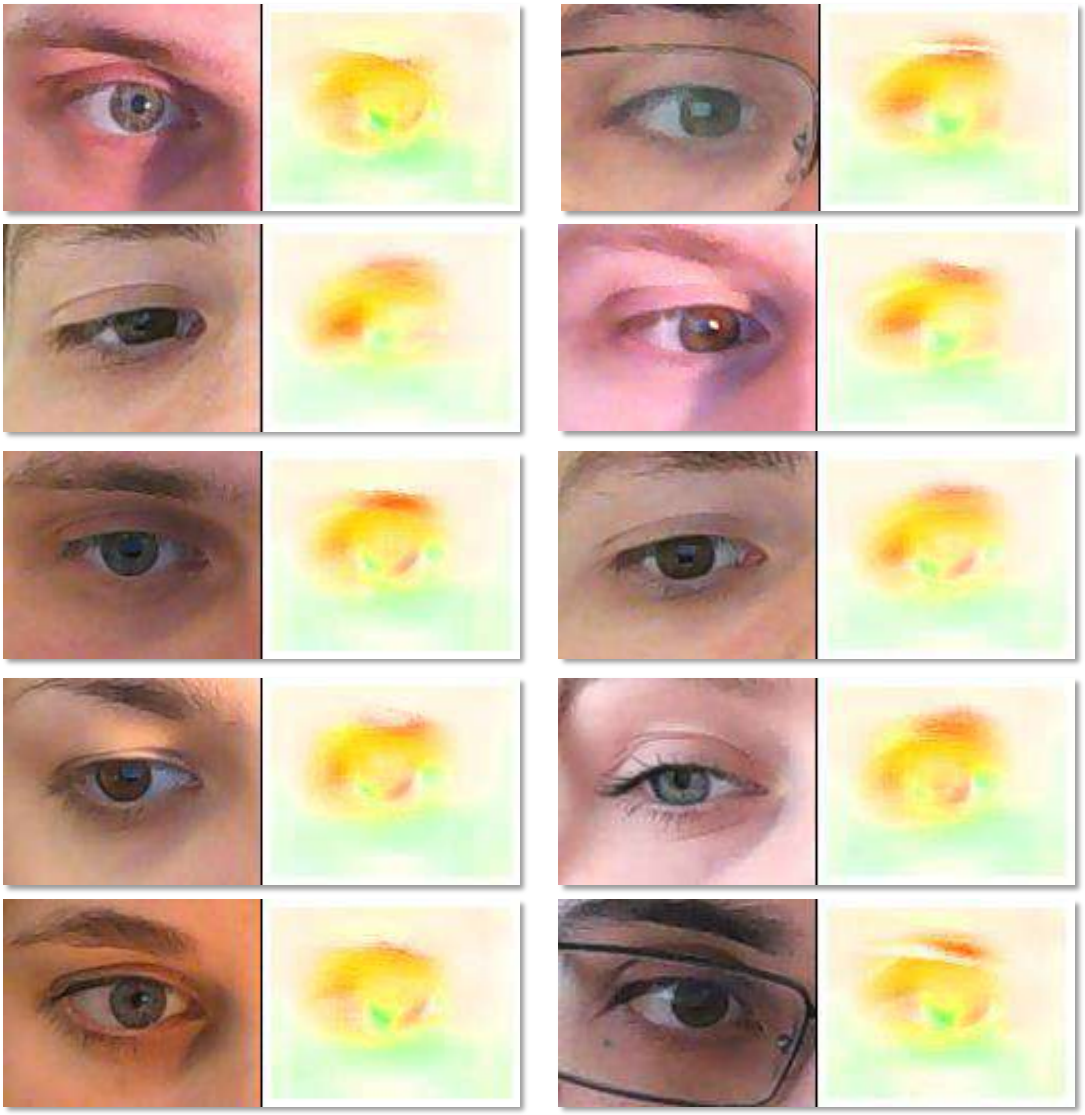


FIGURE 5.5: Examples of warping field taken randomly from a validation set. In each pair, the left image is the input and the right one is the predicted warping field. The color pattern is the same as in Figure 5.1.

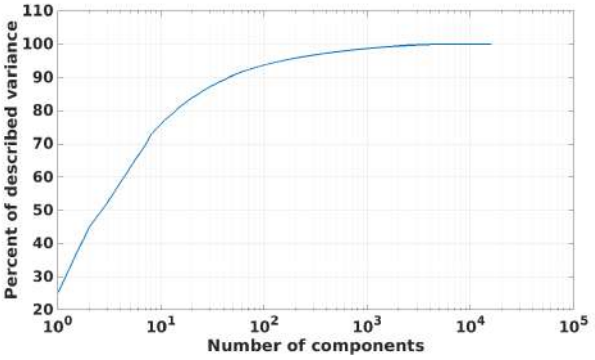


FIGURE 5.6: Inherent dimensionality of the warping fields manifold. The dependence of the percentage of explained variance on the number of PCA components is plotted. The scale of the x-axis is logarithmic. One hundred components describe almost 95% of the variance.

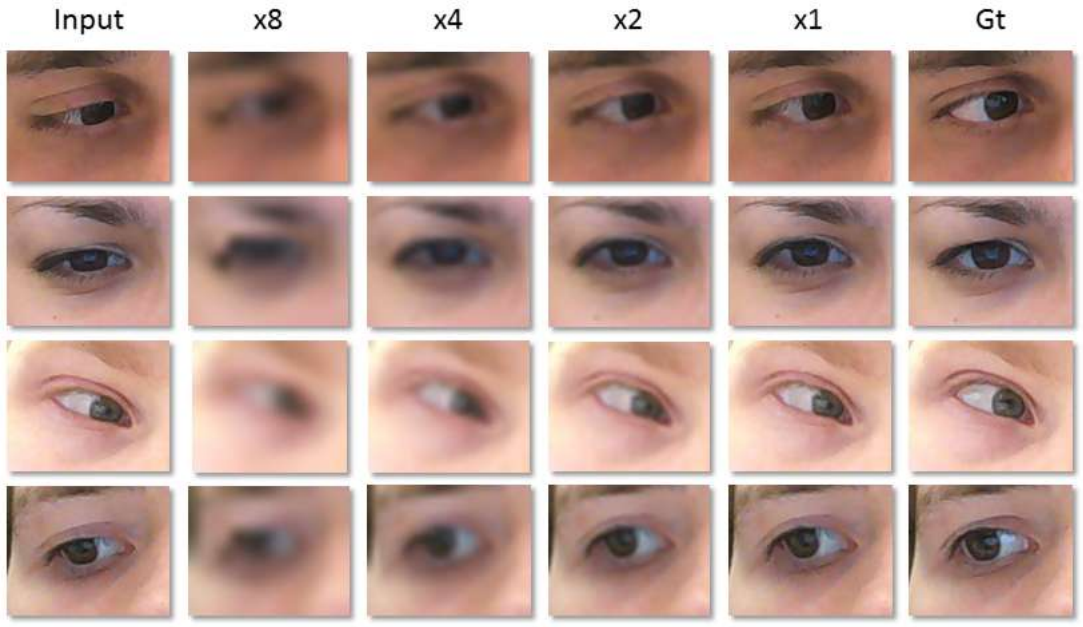


FIGURE 5.7: The results of NNSF system for the 15° upwards redirection with input images of lower resolutions (the downsampling factors are shown at the top). Gaze redirection is persistent even for very low resolution images.

requires a simple per-person pre calibration step and requires a (low-end) GPU for real-time operation (whereas my method achieves more than 30 fps on a single core of an Intel Core-i5 CPU).

Finally, the supplementary video [104] demonstrates a real-time screencast of NNSRF system running on a variety of people under a variety of conditions typical for teleconferencing scenario.

5.3.4 User study

To confirm the perceptual improvement in the results, I performed a user study similar to one presented in Section 4.6.2.1. Here I changed the setup to showing the full face image, while still instructing the users that only eye region will be resynthesized. Showing full face was motivated by more direct measurement of users' perception of the overall realism of the resulting images.

User study includes 41 subjects unrelated to computer vision, comparing four methods (EFF, NNSF, CFW, CFW+LCM). Each user was shown 80 quadruplets of images, and in each quadruplet one of the images was obtained by re-synthesis with one of the methods, while the remaining three were unprocessed real face images. The example screen-shot from the user study interface is shown in Figure 5.9.

FIGURE 5.8: Comparison with monocular gaze correction method from [17]. **Left** – the input video frame (taken from [17]). **Right top** – the output of suggested NNSF system. **Right bottom** – the result of [17]. While both systems achieve convincing redirection effect, NNSF system avoids the distortion of facial proportions (especially in the forehead and the chin regions), while also not requiring GPU to achieve real-time performance.

Twenty randomly sampled results from each of the compared methods were embedded in the set of quadruplets shown to each participant. The ordering of the methods and the position of the right answer were randomized. When a quadruplet was shown, the task of the subject was to click on the artificial (re-synthesized) image as quickly as possible. For each method, I then recorded the number of correct guesses out of 20 (for an ideal method the expected number would be 5, and for a very poor one it would be 20). I also recorded the time that the subject took to decide on each quadruplet (better method would take a longer time for spotting). Table 5.1 shows results of the experiment.

In general, all methods performed very well, approaching the best performance, which is 25% (the performance of random guess). The performances of the NN-supervised random forest and the full deep warp system are comparable, while the other two systems (warping flow forest and coarse-to-fine warping without lightness correction) performed a

	EFF	**NNSF**	**CFW**	**CFW+LCM**
Correctly guessed (out of 20)				
Mean	7.12	5.68	6.16	**5.58**
Std	0.87	0.98	0.92	1.10
Median	**6**	**6**	**6**	**6**
Max	13	10	10	**9**
Min	4	2	**1**	2
Correctly guessed within 4 seconds (out of 20)				
Mean	1.9	**1.66**	1.98	1.85
Std	0.54	0.67	0.59	0.62
Median	1	1	2	**0**
Max	6	7	6	**5**
Min	**0**	**0**	**0**	**0**
Correctly guessed within 2 seconds (out of 20)				
Mean	**0.56**	0.80	1.09	0.96
Std	0.69	0.55	0.48	0.46
Median	**0**	**0**	**0**	**0**
Max	**4**	6	5	**4**
Min	**0**	**0**	**0**	**0**
Mean time to make a guess				
Mean time, sec	7.7	7.3	9.1	**9.7**
Std, sec	1.9	2.5	2.0	2.2

TABLE 5.1: **User assessment for the photorealism of the results for the four methods.** During the session, each user observed 20 instances of results of each method embedded within 3 real images. The participants were asked to click on the re-synthesized image in as little time as they could. The first three parts of the table specify the number of correct guesses (the smaller the better). The last line indicates the mean time needed to make a guess (the larger the better). The performance is not far from the performance of a random guess, thus, re-synthesized images could be hardly distinguished from the real ones.

little worse. However, if considering only fast (confident) clicks, neural networks become slightly worse in comparison to warping flow forest, as well as to NN-supervised forest. Such performance of the deep architecture is explained by the fact, that it works on the fixed basic resolution, while the forest based methods could process images at the eye on native resolution of the cropped image. For a few examples, the users were apparently able to quickly notice the interpolation artifacts or the artifacts near the border in the deep warp system results. As a result, the teacher CFW method performs a bit worse than the student NNSF in these metrics. In terms of mean time that a user took for making a guess, deep warp architectures outperformed forest-based, because of the big contribution of the hardest samples, where users got stuck for a long time. However, one should take into consideration, that the variance in the users results is very high: some people are much attentive than other. Increasing the number of people does not solve the problem, because there is still a significant amount of very good and very bad results. Only the very large number of tests for each user could potentially help, but it's

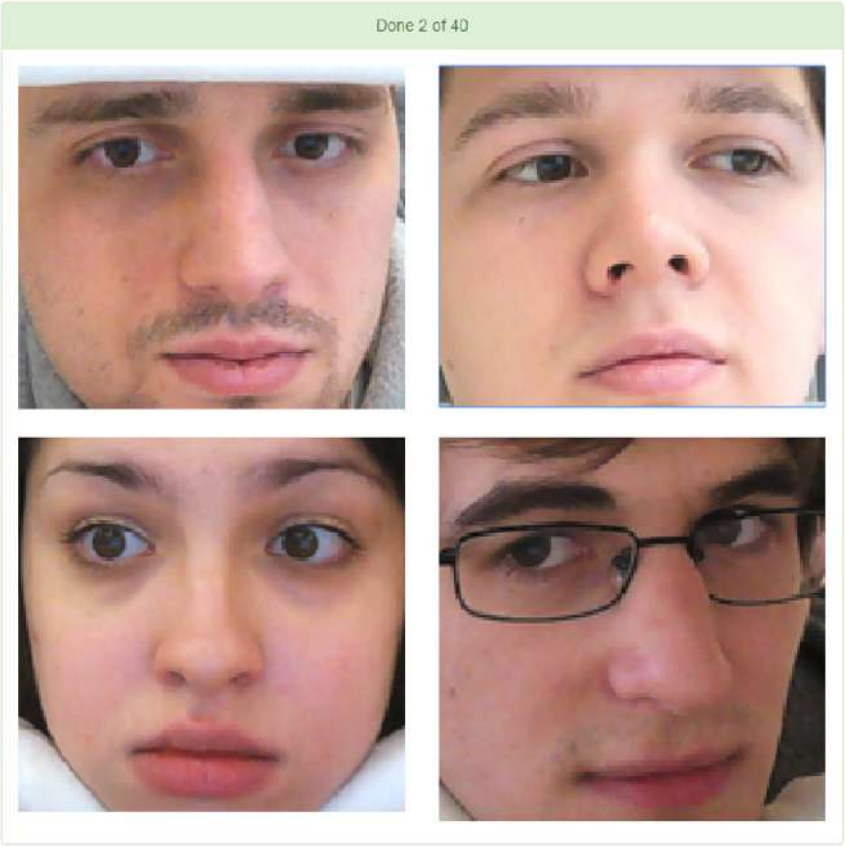


FIGURE 5.9: The screen-shot from the user study interface. User was instructed, that one of the four images is not real, and was asked to click on the one, which seems unnatural, spending not much time (trying not to exceed 5 seconds). In this example, the top left image is the right answer.

hopeless to ask users for several hours of testing without proper motivation. Thus, for example, applying statistical tests to analyze results is difficult.

I also present some comparisons for pairs of methods. At the plots on Figure 5.10 red bar is how many users were fooled more frequently by the first method, green bar by the second method, blue bar draw. NN-supervised random forest is the best method in this comparison.

5.3.5 Computational speed and memory demands

The main testbed for video-conferencing is a standard 640×480 stream from a laptop camera. Facial alignment takes a few milliseconds per frame. On top of the feature tracking time, the warping flow forest method requires from 3 to 30 ms to perform the remaining operations like querying the forest, picking optimal warping flow vectors, and performing replacements. The large variability is due to the fact that the bulk of operation is linear in the number of pixels we need to process, so the 30 ms figure

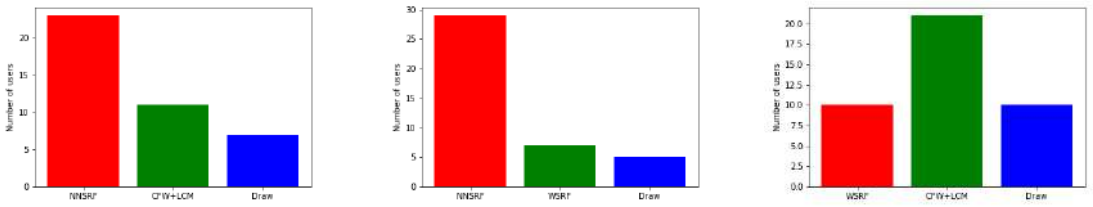


FIGURE 5.10: Results of user study for pairs of methods. Red bar is how many users were fooled more frequently by the first method, green bar by the second method, blue bar draw. For example, in the middle plot forest, supervised by neural network, outperformed weakly-supervised forest.

corresponds to the situation with the face spanning the whole vertical dimension of the frame. Further trade-offs between the speed and the quality can be made if needed (e.g. reducing the number of trees twice will bring only very minor degradation in quality and almost two-fold speedup).

The neural network supervised forest method performs comparably to the warping flow forest, requiring $3 - 30$ ms per frame on single core of a CPU (Intel Core i5 2.6GHz). In fact, it is typically slightly faster than warping flow forest, because of the smaller tree depth, however this difference is not crucial in my implementations. The significant improvement is in memory consumption: warping flow forest method typically requires $100 - 200$ Mb, which should be stored in RAM memory at test-time, while the neural network supervised forest requires only $1.5 - 3$ Mb. The big difference in memory requirements occurs because of error distributions stored in leaves in the former case (Section 3.2) and only two-dimensional flow vector in the latter (Section 5.2).

The computational performance of the deep warp method is up to 20 fps on a mid-range laptop GPU (NVIDIA GeForce-750M), and typically $3 - 5$ times slower on a CPU. A model for the deep warp method is much more compact than for the forest-based one (only 250 Kb in experiments), while also being universal, i.e. not tied to a specific redirection angle.

Chapter 6

Semi-supervised gaze redirection using deep embedding learning

The approaches described in Chapter 3, Chapter 4, Chapter 5 have demonstrated that realistic gaze redirection is possible in monocular setting, i.e. without any additional hardware other than a single camera that is used to acquire the images or videos. The suggested warping-based model however relies heavily on supervised machine learning, and in particular requires a considerable amount of eye images labeled with gaze direction. Acquiring such images is tedious and requires imaging of multiple people in constrained setting (in a literal sense – c.f. Figure 2.5, Figure 2.6).

Here, I extend the warping-based model to unsupervised and semi-supervised settings, where most of the learning happens in an unsupervised way using sequences of eye images of different people with varying and unknown gaze direction. The collection of such data is much easier than in the fully-supervised case. In more detail, the model that is presented here, uses unsupervised manifold learning in order to construct the deep embedding of eye images into a low-dimensional latent space (the *encoder* network) and to learn a *decoder* network that constructs a warping flow field based on the latent representation of two eyes from the same sequence.

Once the unsupervised training is accomplished, the system can redirect gaze of an arbitrary eye image by traversing the manifold in a latent space. In particular, image is mapped to latent space and then latent representation is modified by adding a certain vector in order to estimate the latent representation of the target image. The decoder network can then be used to estimate the warping between the source and the unknown target images. The presented model is similar to the visual analogy making of [18] though it predicts the warping fields rather than the target images directly. The model can use a small amount of supervised data (e.g. a single pair of eye images with known difference

in gaze direction) to estimate the displacements in latent space that are characteristics to certain gaze redirections (e.g. lifting the gaze by 15 degrees upwards as is practical to the videoconferencing scenario).

As I show in the experiments (Section 6.4), the resulting semi-supervised solution achieves convincing gaze redirection, which outperforms in visual quality the fully supervised solution of fully supervised deep warping method Chapter 4 in shortage of training data. Compared to the model of [18], using warping rather than direct re-synthesis ensures high realism of the resulting images and avoids the loss of high-frequency details. Prior to that, in Section 6.1 I discuss related approaches, in Section 6.2 I discuss the bulk of the model and how it can be trained on unsupervised data, and in Section 6.3 I discuss how the training of the model can use a small amount of eye images with known absolute or relative gaze direction.

6.1 Related work

The idea of image analogies goes back to at least [105]. The analogy is defined as a relationship $A : B :: C : D$, spoken as "A is to B as C is to D". The task is to synthesize an unknown image D, given A, B, C. Their approach is first to compute Gaussian Pyramids for given images. Then they construct a pyramid for D in such a way that neighborhoods are similar as the on generated from B, and at the same time have similar multiresolution appearances at corresponding locations in A and C. They apply this approach t0 image filtering, texture synthesis and transferring, super-resolution.

My model is related to the more general deep-analogy making model of [18]. The work is utilizing the ability of analogy-making by addition and subtraction Figure 6.1, which was applied to word embeddings in [106, 107]. The model in [18] is trained on tuples (a, b, c, d), such as $a\ ?\ b :: c\ ?\ d$, in test-time their model is capable to produce an unknown d – the result of applying the transformation $a : b$ to c. Encoder $f : \mathbb{R}^D \to \mathbb{R}^K$ and decoder $g : \mathbb{R}^K \to \mathbb{R}^D$ are parametrized as deep convolutional neural networks, trained end-to-end in the whole architecture. The idea is to represent the target transformation by vector $(f(b) - f(a))$ in the embedding space. Applying transformation to a query c is represented by addition this transformation vector to $f(c)$.

The suggested objective for addition-based analogy making is

$$L_{add} = \sum_{a,b,c,d} ||d - g(f(b) - f(a) + f(c))||_2^2. \tag{6.1}$$

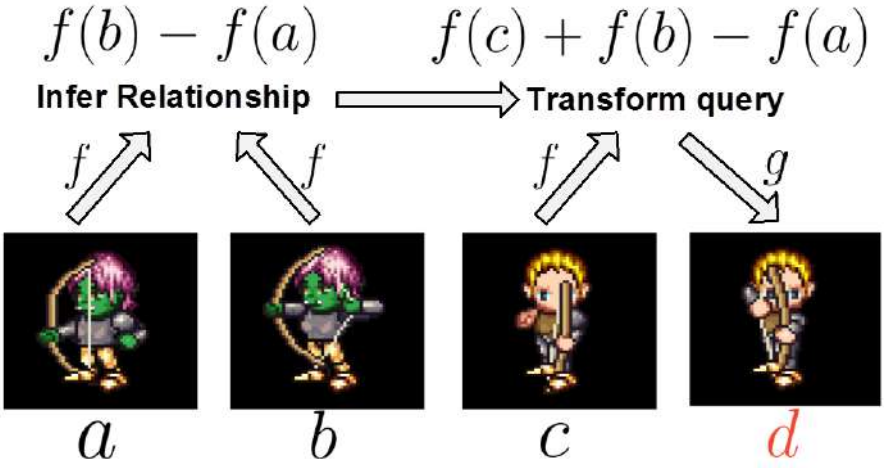


FIGURE 6.1: **The concept of visual analogy making.** An encoder function f maps both query image and an analogy pair into an analogy-making space. In this space analogy is some simple operation (e.g.) summation. Decoder retrieves the result back to image space. Figure taken from [18].

As vector-addition approach is sometimes not capable to model all necessary transformations (e.g. rotation: adding the same vector multiple times will not return to the original embedding vector), [18] also suggests multiplicative and deep model:

$$L_{mul} = \sum_{a,b,c,d} ||d - g(W \times_1 [f(b) - f(a)] \times_2 f(c) + f(c))||_2^2, \tag{6.2}$$

$$L_{deep} = \sum_{a,b,c,d} ||d - g(h([f(b) - f(a); f(c)]) + f(c))||_2^2, \tag{6.3}$$

where $W \in \mathbb{R}^{K \times K \times K}$ is a 3-way tensor and $h : \mathbb{R}^{2K} \to \mathbb{R}^K$ is an MLP. For accurate traversing image manifolds an the embedding space, [18] suggests a regularization, which compares the difference of embeddings of prediction and query $f(d) - f(c)$ to the predicted increment vector in embedding space $T(f(a), f(b), f(c))$:

$$R = \sum_{a,b,c,d} ||f(d) - f(c) - T(f(a), f(b), f(c))||_2^2,$$

where increment $T(f(a), f(b), f(c))$ is one of the three chosen models: additive (6.1), multiplicative (6.2) or deep (6.3).

Authors of [18] also suggest a method to incorporate features disentangling into the analogy model. The model is learned on three-image tuples (a, b, c): the first and the second to extract hidden units in latent representation, and the third one is a ground-truth image. Disentangling is implemented by introducing the binary vector of switches $s \in \{0, 1\}^K$. It represents whether to take the corresponding feature from the embedding of the first image or the second one.

In comparison to my work, the model [18] requires the knowledge of transformations that needs to be provided in the form of analogy-forming quadruplets, which is not required by my model. A more similar to my model at train time is the inverse graphics model of [19], which can be trained with a similar level of supervision to the system suggested here, i.e. with subsets of images where some factors of variations are fixed while others are varying arbitrarily. Authors also makes use of autoencoder-like architecture with image embedding latent space. They present an approach for learning interpretable latent space representations for out-of-plane rotations and lighting variations. Their training procedure is constructed to encourage each group of variables in the hidden representation to distinctly represent some specific transformation.

The latent space is divided into extrinsic variables (elevation angle, azimuth angle, angle of the light source) and intrinsic properties, which describes the identity, shape, expression, etc. The data is organized into mini-batches in such a way, that only one of the extrinsic variables is changing across the mini-batch, all other extrinsic and intrinsic variable being fixed, e.g. rotation of the same object. The training procedure is as follows.

1. Randomly select an extrinsic variable z_{train} and a mini-batch in which only this variable changes.
2. Get the latent representation of each example in the mini-batch using the encoder network.
3. Calculate the mean of those representation vectors over the mini-batch.
4. For all examples in the mini-batch replace all latent variables except chosen z_{train} with the average across the mini-batch.
5. Project mini-batch back to the image space using the decoder, calculate reconstruction error and backpropagate it through the decoder.
6. Replace the gradients for all latent variables except chosen z_{train} with the difference between the values of variables and their mean value across the mini-batch.
7. Backpropagate through the encoder.

In this procedure, step 3 ensures that all the variation in the chosen factor is kept in the only one chosen variable z_{train}. Step 6 makes all the other variables be the same across the mini-batch.

Both [18] and [19] however tend to produce blurry non-photorealistic images that are not suitable for the application scenarios of gaze redirection Figure 6.2. This is overcome in the model suggested in this chapter by using warping instead of direct re-synthesis.

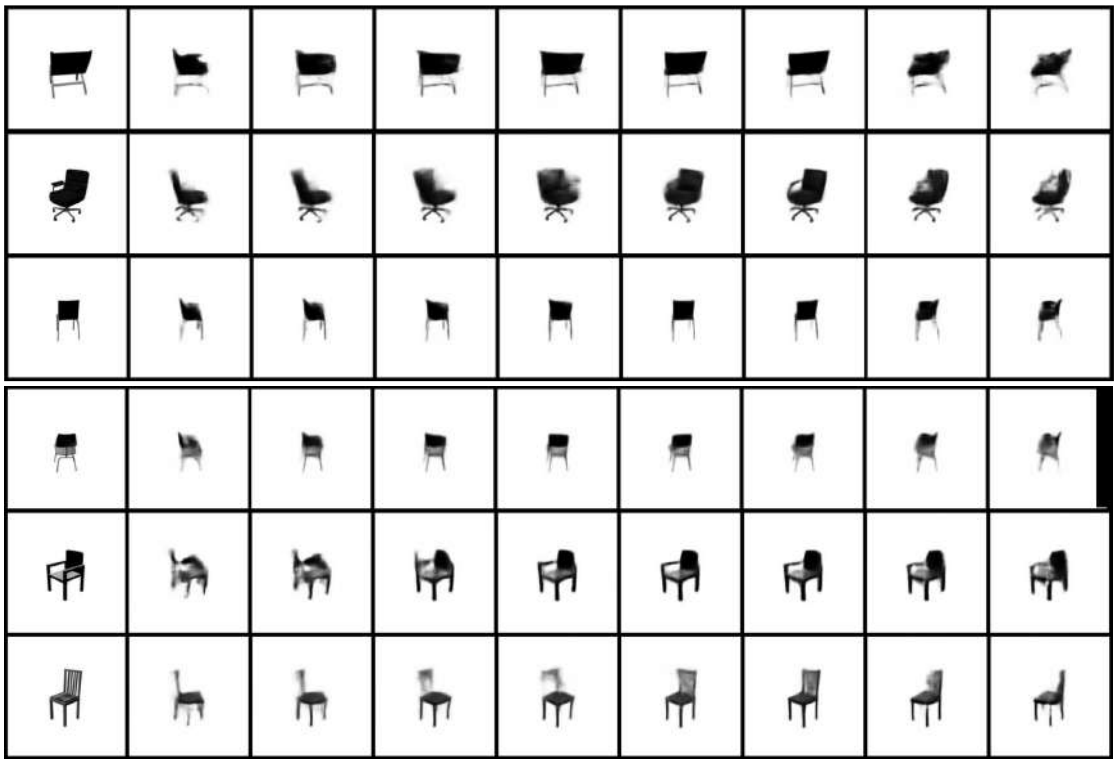


FIGURE 6.2: **Manipulation rotation results from [19].** Each row was generated by encoding the input image (leftmost) with the encoder, then traversing the manifold by changing the value of a single latent and putting this modified encoding through the decoder. Despite the rotation is quite convincing, obvious blurring and a loss of fine details are the drawbacks of the suggested approach in comparison to a warping model (as well as in Figure 2.2, Figure 2.3).

6.2 Unsupervised training of gaze redirection

The process of supervised data collection and eye localization is described in Section 2.3. For an eye localization, I use definition (2.8). As discussed below, located eye landmarks are also embedded in the architecture as additional features in the same way, as in Deep Warp architecture. Thus, all input images I mentioned here are actually not 3 RGB maps, but 17 maps, 14 of which come from landmarks (details are given in Section 4.3). The one exception is final warping, which is applied to an RGB image.

For the unlabeled part of dataset the process is significantly simplified. The person is instructed to keep the head approximately still and to quickly move the gaze along the screen for about 10 seconds. This recording time is comfortable for not blinking and not shaking head, while sufficiently long for a person to gaze at different parts of the screen. This scenario also eliminates the problems with the person not following the dot on the screen as prescribed, which I found out to be a recurrent problem.

Model architecture. I now discuss the architecture of the suggested approach (Figure 6.3) as well as the training of the encoder and the decoder networks, which, as

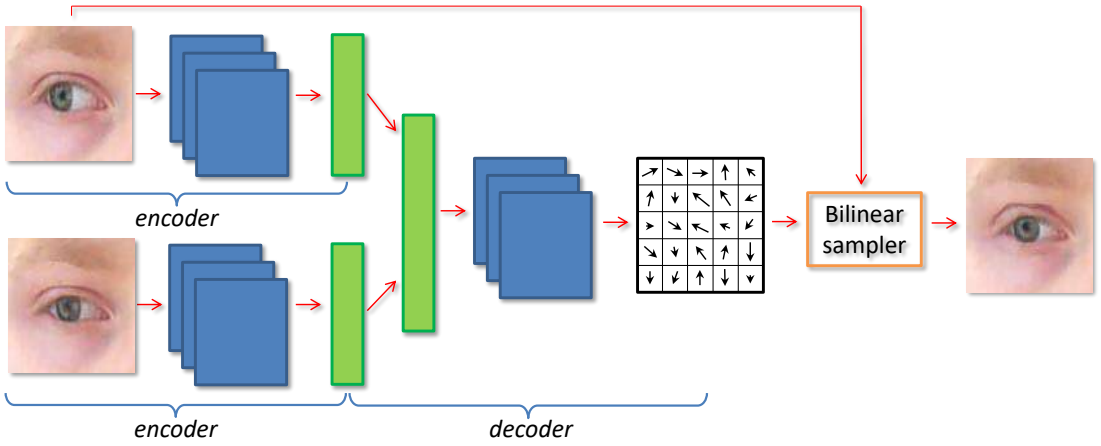


FIGURE 6.3: Architecture of unsupervised gaze redirection training. Two images of the same eye with different gaze direction are passed to the *encoder* network that outputs their latent representations. These representations are then concatenated and passed to the decoder network that outputs the predicted flow from the first image to the second one. The flow can be used by the bilinear sampler. The architecture is trained by minimizing the disparity between the output image and the second image.

discussed above, happens in unsupervised mode and utilizes only image sequences with varying but unknown gaze direction.

In general, similarly to approaches in previous chapters, the gaze redirection is performed by warping the input eye images. At the core of suggested system is the ability to model the change of appearance within the pair of the eye images (I_1, I_2) from the same video sequence using warping. Such warping is determined by the *latent* representations of the images $h_1 = \mathbf{E}(I_1; \psi)$, $h_2 = \mathbf{E}(I_2; \psi)$, where $\mathbf{E}$ denotes a feed-forward *encoder* network with learnable parameters ψ. The latent representations live in low-to-medium dimensional space (up to 500 dimensions in experiments, which is much smaller than the dimensionality of the original images).

Given the latent representations of the images, a *decoder* network $\mathbf{D}$ with learnable parameters ω is applied to the stacked latent representations of the image pair and outputs the warping field, corresponding to the transformation of image I_1 into I_2: $\mathcal{F} = \mathbf{D}(h_1, h_2; \omega)$.

Finally, the standard bilinear sampling layer $\mathbf{S}$ as defined in [82] outputs the result, which is the prediction $\hat{I}_2$ of image I_2. Overall, the warping process can be written as:

$$\hat{I}_2(I_1; \psi, \omega) = \mathbf{S}(I_1, \mathbf{D}(\mathbf{E}(I_1; \psi), \mathbf{E}(I_2; \psi); \omega))). \tag{6.4}$$

The training objective then naturally corresponds to minimizing the disparity between the true image I_2 and its predicted version (6.4). The training process then corresponds to sampling pairs $(I_1; I_2)$ and optimizing the parameters for the encoder and the decoder

networks by minimizing the following ℓ_2-loss:

$$L(\psi, \omega) = \sum_{(I_1, I_2)} \|\hat{I}_2(I_1; \psi, \omega) - I_2\|^2 ,$$

where the summation is taken over all training pairs of eye images that correspond to the same sequences.

Notably, the training process does not require gaze annotation, and, as will be verified below, learns meaningful latent representations that are *consistent* across eye sequences in the following sense. Let a *visual analogy* be quadruplet (I_1, I_2, I_3, I_4), in which I_1 and I_2 correspond to one eye sequence, and I_3 and I_4 correspond to other eye sequence (corresponding to a potentially different person and/or different lighting etc.), and where the change of gaze direction from I_1 to I_2 and from I_3 to I_4 are similar Figure 6.5. The learned embeddings possess the property of having similar displacement vectors across the two pairs:

$$\mathbf{E}(I_1; \psi) - \mathbf{E}(I_2; \psi) \approx \mathbf{E}(I_3; \psi) - \mathbf{E}(I_4; \psi) \tag{6.5}$$

The property (6.5) facilitates easy semi-supervised training of the overall system with limited amount of gaze-annotated data.

Details of the architectures.

To describe the specific architectures, I denote $\text{conv}(m, k, s)$ a convolutional layer with m maps, kernel size k and size of the stride s, and $\text{FC}(m)$ a fully connected layer with m maps, both precede the RELU activation. The architecture of the encoder I used in experiments is the following:

$$\text{conv}(48, 5, 1) \to \text{conv}(48, 5, 2) \to \text{conv}(96, 5, 2) \to \text{conv}(96, 3, 2) \to FC(800) \to FC(50).$$

The decoder mirrors the architecture of the encoder, except for the input (which is the vector of length 100, being a concatenation of two representations of length 50) and the output, which are two maps of the warping field used in (6.4). The model is trained using Adam optimizer [100]. Each batch contains 128 randomly sampled pairs of images, each pair consisting of the input and output eye from the same sequence.

6.3 Semi-supervised learning and analogies

The architecture discussed above trains on pairs of eye images, and treat each of the images in the pair similarly. At test time, however, the goal is to compute the warping field *without* knowing the second image (which itself is the unknown that one wish to

estimate). Fortunately, the analogy property (6.5) possessed by the embeddings allows us to estimate characteristic displacements in the latent space given some amount of gaze direction-annotated data obtained with the time-consuming process.

I consider the following test-time gaze redirection problem: given the query image I_q, obtain the image O^q corresponding to the same eye under the same imaging condition, with the gaze redirected by a given angle $\alpha_q = (\alpha_q^x, \alpha_q^y)$. As the angle α_q is given, we can query the direction-annotated part of the dataset for the set of pairs $P(\alpha_q) = \{(I_1^1, I_2^1), \dots, (I_1^n, I_2^n)\}$ that would form an analogy with I_q and O_q, i.e. the pairs with the difference in the gaze direction within each pair approximately equal α_q (in practice I use a hard threshold ϵ to determine whether some angular difference is close enough to α_q).

I then consider two methods of computing O_q given the set of pairs $P(\alpha_q)$. The first (baseline) method is to use the *mean warping field* of the set of analogy pairs. Here, for each pair I calculate the predicted warping field from the first image in the pair to the second, and then apply the averaged warping field $\bar{\mathcal{F}}$ to the query image:

$$\begin{aligned} &\hat{O}_q = \mathbf{S}(I_q, \bar{\mathcal{F}}), \quad \text{where} \\ &\bar{\mathcal{F}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{D}(\mathbf{E}(I_1^i; \psi), \mathbf{E}(I_2^i; \psi); \omega). \end{aligned} \tag{6.6}$$

However, the clear drawback of this method is that the same warping field $\bar{\mathcal{F}}$ will be applied to all query images with the same desired angular redirection α_q, being independent from the content of the query I_q.

The second method that directly relies on the analogy property (6.5) computes the mean latent vector displacement corresponding to angle α_q (Figure 6.4):

$$\Delta h(\alpha_q) = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{E}(I_2^i; \psi) - \mathbf{E}(I_1^i; \psi)).$$

Such precomputed vector can be used to estimate the desired output image as:

$$\hat{O}_q = \mathbf{S}(I_q,\ \mathbf{D}\left(\mathbf{E}(I_q; \psi),\ \mathbf{E}(I_q; \psi) + \Delta h(\alpha_q); \omega\right)). \tag{6.7}$$

All latent representations for labeled part of the dataset could be precomputed in advance and stored. Thus, performing the redirection following (6.7) requires only a single pass through the encoder and the decoder network at test time, which opens up a possibility for real-time gaze manipulation (on a device with a GPU).

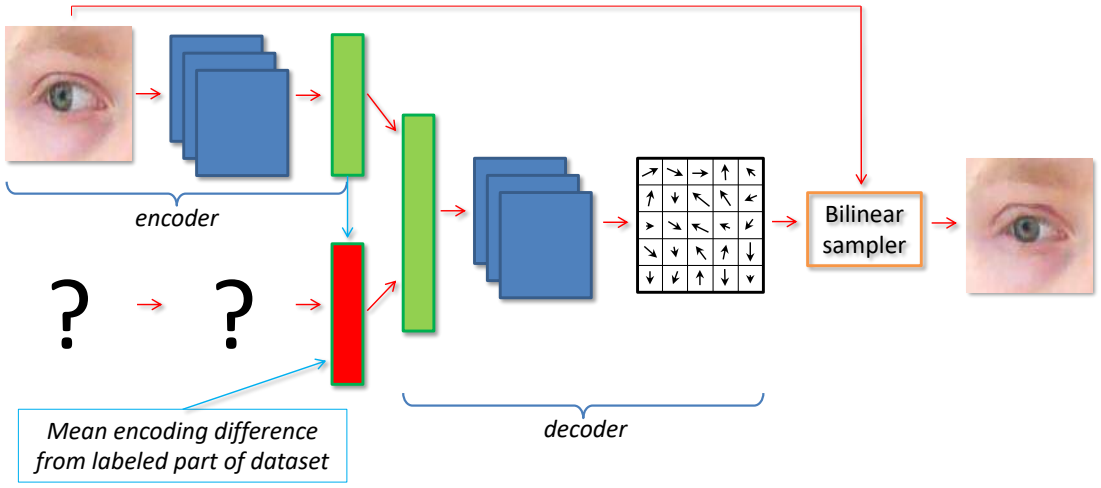


FIGURE 6.4: Image analogy-making in test-time. In test time we do not know second image, we want to produce it. Thus, we do not have its representation for the encoder. The solution is to approximate the unknown representation in the latent space. The estimate of the unknown embedding is the known representation plus mean increment, which we get from the labeled part of the dataset.

6.4 Experiments

I perform experiments using the dataset that consists of 640 sequences, each containing images of the same eye from one video (under the same lightning conditions, head pose, etc.) with different known gaze directions. Each sequence contains $100 - 220$ images. I use 500 sequences for training and validation, leaving 140 for testing (the train and the test sets do not contain sequences of the same people). The angular tolerance ϵ for picking up analogies from the labeled part of dataset was set to $0.5°$.

Qualitative evaluation of unsupervised learning via analogies. I first qualitatively demonstrate the analogy property (6.5) in Figure 6.5. For each input query image and a query redirection angle, one analogy quadruplet with the similar angular difference was picked up at random. The fourth image of the quadruplet was then obtained using the method that modifies the representation in latent space (6.7) by warping the third image and using the first two images as the only reference pair. The obtained results mostly look plausible and similar to the ground truth, confirming the analogy consistency of the learned embedding. Note, that except for angular difference in gaze direction, all other properties could be different between the two pairs. This includes the absolute gaze direction, the identity of the person, the lightning conditions, the presence of glasses, etc.

Quantitative evaluation of semi-supervised learning. I then perform quantitative evaluation for the task of fixed redirection angle $15°$ upwards. I consider the following methods:

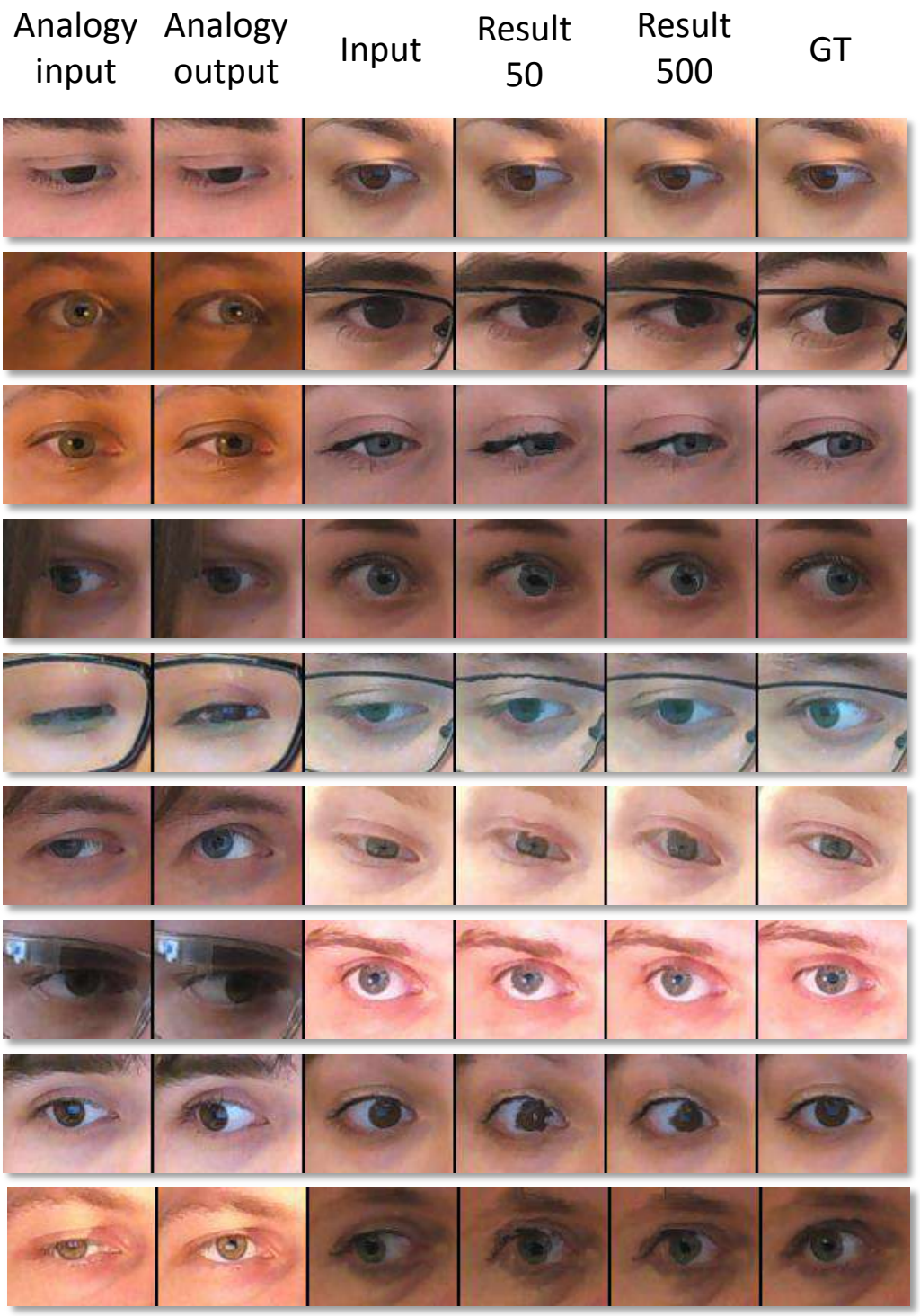


FIGURE 6.5: **Demonstration of analogy property of the learned embedding.** The three left-most columns alongside the right-most one form analogy quadruplets (the difference in gaze direction between the first two columns is approximately the same as the difference in gaze direction between the third and the last columns). 'Results 50' and 'Results 500' demonstrate the warped images obtained using modification in latent space (as discussed in the text), after the encoder and the decoder are trained in an unsupervised setting on 50 or 500 eye sequences respectively. Failure cases with dramatical eyeball deformations are presented at bottom rows. Overall, training on more unsupervised data (500 sequences) leads to better and more robust result.

- The unsupervised system trained on different amount of unlabeled sequences, as discussed in Section 6.2, with two approaches for test-time prediction:
  1. Based on mean displacement vector in representation space (6.7), denoted as "code".
  2. Based on mean warping field (6.6), denoted as "flow".
- The single-scale DeepWarp system (2), denoted as "SS". I use the single scale version as a baseline since the architecture of the flow-predicting network in single-scale DeepWarp is similar to encoder-decoder network suggested here in complexity. Note that the encoder-decoder network architectures presented here also allow multiscale extensions.

During training time, all methods were trained for the task of redirection by an arbitrary angle (the redirection angle was fixed for the testing only). I vary the amount of labeled sequences shown to the methods. The unsupervised models were trained for 150 epochs on the unlabeled datasets containing either 100 or 500 sequences. For images from test set, I pick all possible analogies from given labeled sequences, and vary the number of sequences in this labeled part. The DeepWarp system requires full supervision and therefore was trained only on the labeled part of the dataset for 150 epochs. The quantitative comparison is represented in Figure 6.6. I evaluate the mean (over pixels) sum of squared errors across channels (referring to it as MSE). The semi-supervised models outperform the DeepWarp model, which does not exploit the unlabeled data, and, as expected, the advantage is bigger when the amount of labeled data is smaller and when the size of the unlabeled dataset used within the semi-supervised model is bigger. Increasing the number of unlabeled sequences improves the performance of the model. The method based on latent representation (6.7) better exploits the trained unsupervised model, than the baseline which averages the warping flows. The performance of semi-supervised methods saturate after seeing approximately 15 labeled sequences.

For the reference, I also measured the MSE of two more baselines. The *Unsupervised oracle* baseline corresponds to the unsupervised model that knows the latent representation of the ground truth and uses it to estimate the warping field. This oracle baseline bounds the performance of semi-supervised methods from below and achieves the MSE 0.0033 for 100 sequences of unlabeled data and 0.0025 for 500 sequences. On the other hand, the MSE of leaving the input image unmodified was 0.0073.

Qualitative evaluation of semi-supervised learning. Finally, I demonstrate the qualitative results of redirection on arbitrary angles in Figure 6.7. All systems use 15 labeled data sequences. The semi-supervised model is trained on 500 sequences of unlabeled data. Performing analogies in the latent representation space allows to get a

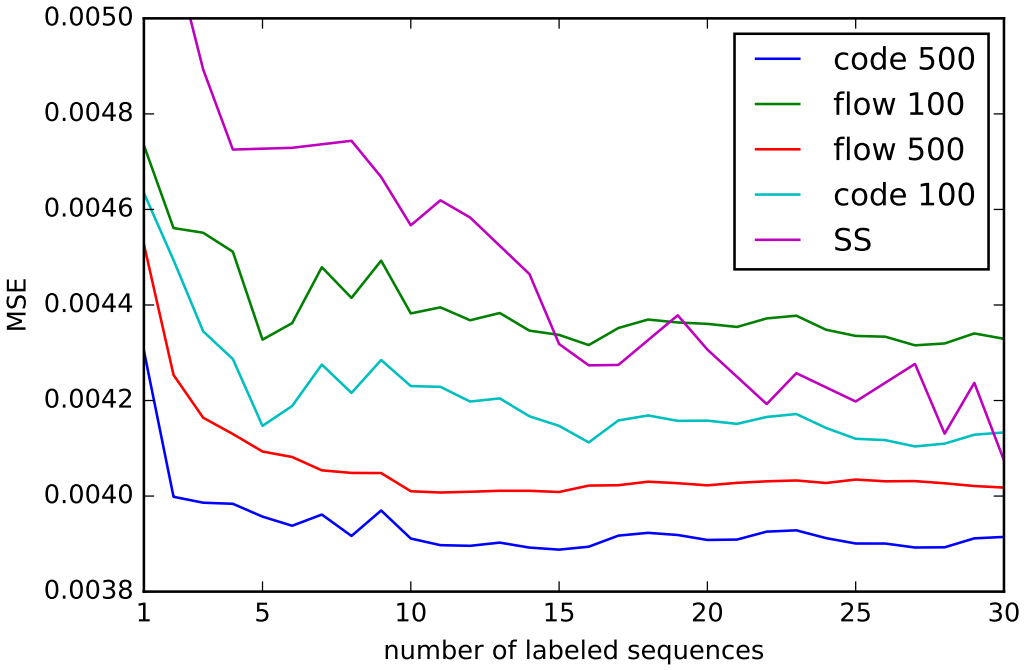


FIGURE 6.6: **Quantitative comparison of methods: errors for redirection on 15 degrees upwards.** Horizontal axis shows the number of sequences labeled with gaze direction provided to the methods. All methods are trained for arbitrary redirection angle, but applied to a testing setting with 15 degrees vertical redirection. "Code" corresponds to estimating the warped image latent representation (6.7), "flow" corresponds to warping the input image using mean flow (6.6). 100/500 corresponds to the number of unlabeled sequences used to train the encoder-decoder model. "SS" stands for the single scale DeepWarp system that does not use unlabeled data. Semi-supervised system performing analogies in latent representation space outperforms other methods, and training this method on more unlabeled data helps a lot irrespective the amount of labeled data.

substantial perceptual improvement over the results of supervised model in the lack of training data.

I also present the results of a vertical redirection for the range of angles in Figure 6.8 for the semi-supervised method based on analogies in the latent representation space (6.7) trained on 500 unlabeled sequences and using 15 labeled sequences for performing analogy-making.

FIGURE 6.7: **Sample results on the hold-out set.** Columns from left to right: the input image, the ground truth, the results of single scale deep warp system, the result of the semi-supervised model that uses mean warping field, the result of the semi-supervised model that uses mean difference in latent representation space. With limited labeled data the perceptual quality of the results is significantly improved using large dataset of unlabeled data. Failure cases with very extreme redirection angles and glasses damage are presented in six bottom examples.

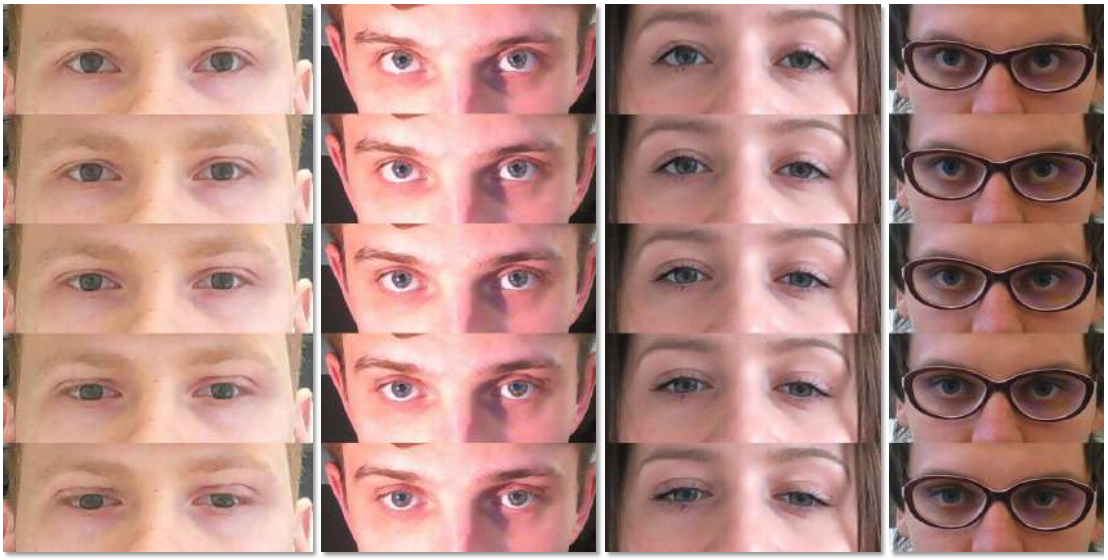


FIGURE 6.8: **Vertical redirection.** The redirection angle ranges from $-15°$ to $15°$ relative to the central input image. In all columns the image in the center row is the input. Convincing continuous gaze redirection is achieved.

Chapter 7

Conclusions and summary

The thesis has presented the warping-based approach to image re-synthesis task, where the target transformation is defined by a dataset of input-output examples, and its application to the gaze redirection task. This approach increases the photorealism of the results in comparison to the direct regression of output pixels, which has problems with synthesizing high-resolution images. The work [44] and similar works suggested to construct the warping field based on geometrical projection properties, but the idea of learning the warping field from the dataset of examples to the best of my knowledge was not presented until this work [73].

The four methods for learning a warping field predictor are suggested. Below, I once again summarize the suggested methods, and state their novelty compared to the most similar preceding works in the literature. The three of the methods learn to predict the warping field in a weakly supervised setting. The first system (Chapter 3, [73]) is based on a special kind of weakly supervised random forest with structured output. The novelty compared to the preceding work [76] is that not only the optimal warping vector, but the distribution of the replacement error depending on the warping vector is stored in the leaves of the tree. Storing these distributions also allows more robust tree ensembling by summing up distributions coming from different trees. Also, the split evaluation criterion is new and specific to the newly proposed type of random forest. The second system (Chapter 4, [74]) predicts the warping field using a deep convolutional network with coarse-to-fine architecture of warping modules and embeds the desired redirection angle and feature points as image-sized maps. In addition to warping, photorealism is increased using the lightness correction module. The third system (Chapter 5, [75]) can be regarded as a hybrid of the first two, as it essentially condenses the neural network into a random forest, achieving high-quality results, fast operation, and very compact models.

The novelty of the suggested architecture compared to the preceding work on teacher-student architectures [77–79] is in the particular teacher-student combination, where the teacher is a weakly supervised neural network, and the student is a random forest. Such combination allows to combine the representation power of the deep neural network in the fast forest model. The fourth system (Chapter 6) is motivated by the hardness of collecting labeled database and is working in a semi-supervised scenario, requiring only small labeled part in the dataset. It is based on learning the embedding of images into a low-dimensional latent space and analogy making in this latent space. The novelty of the method in comparison to work [18] is in using the combination of unsupervised training on pairs of images without gaze labels, instead of requiring analogy-forming quadruplets, and image analogy making at test time. Another peculiarity of suggested method is that the result of analogy-making is a warping-field rather than a re-synthesized image itself. This, once again, allows to boost the photorealism of the result.

In application to gaze redirection, the common approach of all methods is to resynthesize the area of eyes in order to emulate the change in gaze direction without changing the head pose. The two forest-based systems redirects the gaze by a fixed angle and runs in real-time on a single CPU core. The Deep Warp system take a redirection angle as an input and thus allows to change gaze continuously in a certain range, while also obtaining the higher quality result. The semi-supervised system also redirects gaze by an arbitrary input angle.

The learned model, applied to some image re-synthesis task, is capable to choose the appropriate transformation from some family, provided there is one, which is capable to transform input image to output. In application to warping, this means, that pixels of the output image should be contained in the input image. The gaze redirection task almost satisfies this requirement. The only issue is the lack of white, and the lightness correcting module (Section 4.4) addresses this issue. For image re-synthesis tasks with more serious color changes from input to output image, a probable approach could be to develop more sophisticated refinement network, than brightness correction. The general case is to add refinement without any restrictions. However, depending on the special task, some restricted refinement could be useful, such as the lightness correction module only could increase the brightness of pixels. In the case of gaze redirection, this limited capacity was mostly sufficient for modeling what was not possible to model with warping.

7.1 Discussion

The proposed approaches have some common limitations. The initial idea of resynthesizing the eye vicinity could fail for a very large angles – imagine redirection by 90° – any

attempt to do it only with the ones' eyes without turning the head would look unnatural. However, for small angles, at least in the range $\pm 30^\circ$, the results looks photorealistic provided the eye regions were synthesized naturally.

The collection of the large database allows to rely on machine learning in order to learn a target transformation, which was not presented in the literature on gaze correction before this work. One of the crucial choice in all models was the direct re-synthesis versus the warping based approach. The first approach is to regress pixels of the target image directly. In the experiments with gaze correction data this approach leads to significant blurriness and the loss of fine details. These conclusions correlate with the results reported in the papers [18, 19, 24, 43]. In general, warping-based re-synthesis allows to obtain sharp images. It is however limited in its ability to synthesize realistic dis-occluded areas. The key insight of this thesis is that such warping field can be predicted using a predictor trained in a weakly-supervised or semi-supervised fashion. While being possibly too restrictive for some applications, the warping-based approach happens to work better than the direct re-synthesis in the gaze redirection application. The dis-occluded regions could be also refined by some post-processing models, such as those proposed in this thesis and the one that appeared in the paper published simultaneously with this work [46].

The two types of predictors used in this work are random forests and convolutional neural networks. In general, neural networks obtain better results, both quantitatively and qualitatively, due to their representation power. It coincides with recent advances of convolutional neural networks in the whole computer vision domain. However, random forests are faster predictors due to their "divide and conquer" approach. The methods based on random forests are therefore easy to implement in real time on a single CPU core. "Teaching" a random forest architecture by a neural network allows to obtain the same speed and approach the quality of the network-based model. I consider this solution to be the most suitable for gaze correction in videoconferencing among all solutions proposed in this work

The key advantages of the proposed methods is their ability to work with a monocular input and in real-time on consumer-grade devices. The suggested systems are reasonably robust in the presence of head pose variation and deal correctly with the situations where a person wears glasses.

The drawback of forest-based methods the way they are realized now is the fixed redirection angle. The neural network models, in contrast, take redirection angle as an additional input and are thus capable to produce images with gaze in different directions. The limitation of forest-based models is not critical for a videoconference application,

where a single correction angle which compensates for the vertical gap between the camera and the screen is typically enough. Several models with a range of vertical redirection angles working for different distances from a person to the screen are also a possible solution if flexibility in redirection angle is needed, which one could implement with a real-time performance on a consumer device without a GPU much easier, than designing a fast neural network architecture that can run at multiple frames-per-second on a CPU. However, for other applications, like image and video post-processing, one should be able to redirect gaze by an arbitrary angle. On the other hand, in such applications the requirements on the running speed are less stringent. Thus, for this applications, the Deep Warp system based on deep neural networks is a preferred solution.

The models mentioned above require dataset with fully labeled gaze redirection. Such dataset is difficult to collect. Therefore, the semi-supervised approach for training a neural network model was suggested. In the presence of large amount of such unlabeled data it shows significant improvement over purely supervised models that ignored eye images with unknown gaze direction. The process of data labelling is often the most expensive in computer vision applications. In the particular case of data collection for this work, the necessity for labeled data was the major obstacle. Thus, the opportunity to switch to semi-supervised approach is very useful for large datasets.

Finally I note that reconstruction error is not enough to make a fully justified conclusion about the perceptual quality of the results and can not replace user studies in problems like gaze redirection. Some methods that showed slightly worse results in terms of the ℓ_2 reconstruction error are still perceptually good enough for users. The user study showed the high photorealism of suggested methods, as the guess ratio is close to a random guess. In general, the user studies show the ability of the proposed methods to produce sufficiently realistic results despite the very high standards imposed on the realism of face and eye images by the human visual perception system. The suggested methods were recognized both in the academic society, resulting in several publications, and in the industry, with the commercialization process ongoing and the license purchased by a major company.

Bibliography

[1] Paul Upchurch, Jacob Gardner, Kavita Bala, Robert Pless, Noah Snavely, and Kilian Weinberger. Deep feature interpolation for image content changes. *arXiv preprint arXiv:1611.05507*, 2016.

[2] Ziwei Liu, Raymond Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. *CoRR*, abs/1702.02463, 2017. URL http://arxiv.org/abs/1702.02463.

[3] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[4] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[6] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[7] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[8] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, pages 286–301. Springer, 2016.

[9] Erroll Wood, Tadas Baltrusaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. Gazedirector: Fully articulated eye gaze redirection in video. *arXiv preprint arXiv:1704.08763*, 2017.

[10] Andrew Jones, Magnus Lang, Graham Fyffe, Xueming Yu, Jay Busch, Ian McDowall, Mark T. Bolas, and Paul E. Debevec. Achieving eye contact in a one-to-many 3D video teleconferencing system. *ACM Trans. Graph.*, 28(3), 2009.

[11] Claudia Kuster, Tiberiu Popa, Jean-Charles Bazin, Craig Gotsman, and Markus Gross. Gaze correction for home video conferencing. *ACM Transactions on Graphics (TOG)*, 31(6):174:1–174:6, 2012.

[12] Lior Wolf, Ziv Freund, and Shai Avidan. An eye for an eye: A single camera gaze-replacement method. In *Computer Vision and Pattern Recognition (CVPR)*, pages 817–824, 2010.

[13] Zhixin Shu, Eli Shechtman, Dimitris Samaras, and Sunil Hadap. Eyeopener: Editing eyes in the wild. *ACM Transactions on Graphics*, 36(1), September 2016.

[14] Wikipedia. Bilinear interpolation — Wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/wiki/Bilinear_interpolation. [Online; accessed 21-September-2017].

[15] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.

[16] Jamie Shotton, Ross B. Girshick, Andrew W. Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. Efficient human pose estimation from single depth images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2821–2840, 2013.

[17] Dominik Giger, Jean-Charles Bazin, Claudia Kuster, Tiberiu Popa, and Markus Gross. Gaze correction with a single webcam. In *IEEE International Conference on Multimedia & Expo*, 2014.

[18] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In *Advances in neural information processing systems*, pages 1252–1260, 2015.

[19] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.

[20] Andrew S Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, 2014.

[21] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

[22] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[23] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[25] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[26] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[27] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1462–1471, 2015.

[28] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2016.

[29] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[30] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer, 2014.

[31] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[32] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[34] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[35] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[36] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[37] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] Zhixin Shu, Ersin Yumer, Sunil Hadap, Kalyan Sunkavalli, Eli Shechtman, and Dimitris Samaras. Neural face editing with intrinsic image disentangling. *arXiv preprint arXiv:1704.04131*, 2017.

[40] Edwin H Land and John J McCann. Lightness and retinex theory. *Josa*, 61(1): 1–11, 1971.

[41] Ravi Ramamoorthi and Pat Hanrahan. On the relationship between radiance and irradiance: determining the illumination from images of a convex lambertian object. *JOSA A*, 18(10):2448–2459, 2001.

[42] Ronen Basri and David W Jacobs. Lambertian reflectance and linear subspaces. *IEEE transactions on pattern analysis and machine intelligence*, 25(2):218–233, 2003.

[43] Amir Ghodrati, Xu Jia, Marco Pedersoli, and Tinne Tuytelaars. Towards automatic image editing: Learning to see another you. *CoRR*, abs/1511.08446, 2015. URL http://arxiv.org/abs/1511.08446.

[44] Steven M Seitz and Charles R Dyer. View morphing: Uniquely predicting scene appearance from basis images. In *Proc. Image Understanding Workshop*, pages 881–887, 1997.

[45] Raymond Yeh, Ziwei Liu, Dan B. Goldman, and Aseem Agarwala. Semantic facial expression editing using autoencoded flow. *CoRR*, abs/1611.09961, 2016. URL http://arxiv.org/abs/1611.09961.

[46] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. *CoRR*, abs/1703.02921, 2017. URL http://arxiv.org/abs/1703.02921.

[47] Chris L Kleinke. Gaze and eye contact: a research review. *Psychological bulletin*, 100(1):78, 1986.

[48] R Wade Wheeler, Joan C Baron, Susan Michell, and Harvey J Ginsburg. Eye contact and the perception of intelligence. *Bulletin of the Psychonomic Society*, 13(2):101–102, 1979.

[49] John E Lochman and George Allen. Nonverbal communication of couples in conflict. *Journal of Research in Personality*, 15(2):253–269, 1981.

[50] Mark Cook and JACQUELINE SMITH. The role of gaze in impression formation. *British Journal of Clinical Psychology*, 14(1):19–25, 1975.

[51] Lisa J Wallis, Friederike Range, Corsin A Müller, Samuel Serisier, Ludwig Huber, and Zsófia Virányi. Training for eye contact modulates gaze following in dogs. *Animal behaviour*, 106:27–35, 2015.

[52] Wikipedia. Uncanny valley — Wikipedia, the free encyclopedia, 2017. URL http://en.wikipedia.org/wiki/Uncanny_valley. [Online; accessed 21-September-2017].

[53] Ken-Ichi Okada, Fumihiko Maeda, Yusuke Ichikawaa, and Yutaka Matsushita. Multiparty videoconferencing at virtual social distance: Majic design. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 385–393, 1994. ISBN 0-89791-689-1.

[54] Ruigang Yang and Zhengyou Zhang. Eye gaze correction with stereovision for video-teleconferencing. In *ECCV (2)*, pages 479–494, 2002.

[55] Antonio Criminisi, Jamie Shotton, Andrew Blake, and Philip HS Torr. Gaze manipulation for one-to-one teleconferencing. In *IEEE International Conference on Computer Vision (ICCV)*, pages 191–198, 2003.

[56] Jiejie Zhu, Ruigang Yang, and Xueqing Xiang. Eye contact in video conference via fusion of time-of-flight depth sensor and stereo. *3D Research*, 2(3):1–10, 2011.

[57] Tat-Jen Cham, Shyam Krishnamoorthy, and Michael Jones. Analogous view transfer for gaze correction in video sequences. In *Seventh International Conference on Control, Automation, Robotics and Vision, ICARCV 2002, Singapore, 2-5 December 2002, Proceedings*, pages 1415–1420, 2002.

[58] B Yip and J. S. Jin. Face re-orientation using ellipsoid model in video conference. In *Proc. 7th IASTED International Conference on Internet and Multimedia Systems and Applications*, pages 245–250, 2003.

[59] Yalun Qin, Kuo-Chin Lien, Matthew Turk, and Tobias Höllerer. Eye gaze correction with a single webcam based on eye-replacement. In *International Symposium on Visual Computing*, pages 599–609. Springer, 2015.

[60] Zicheng Liu, Zhengyou Zhang, Chuck Jacobs, and Michael Cohen. Rapid modeling of animated faces from video. *Computer Animation and Virtual Worlds*, 12(4): 227–240, 2001.

[61] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[62] Jianbo Shi et al. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[63] Jiejie Zhu, Liang Wang, Ruigang Yang, and James Davis. Fusion of time-of-flight depth and stereo for high accuracy depth maps. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[64] Jiejie Zhu, Miao Liao, Ruigang Yang, and Zhigeng Pan. Joint depth and alpha matte optimization via fusion of stereo and time-of-flight sensor. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 453–460. IEEE, 2009.

[65] David T Guinnip, Shuhua Lai, and Ruigang Yang. View-dependent textured splatting for rendering live scenes. In *ACM SIGGRAPH 2004 Posters*, page 51. ACM, 2004.

[66] Jason M Saragih, Simon Lucey, and Jeffrey F Cohn. Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision*, 91 (2):200–215, 2011.

[67] Olga Sorkine. Laplacian mesh processing. In *Eurographics (STARs)*, pages 53–70, 2005.

[68] Alan L Yuille, Peter W Hallinan, and David S Cohen. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2): 99–111, 1992.

[69] Mark Everingham, Josef Sivic, and Andrew Zisserman. Hello! my name is... buffy–automatic naming of characters in tv video. In *BMVC*, 2006.

[70] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *Computer Vision and Pattern Recognition (CVPR)*, pages 532–539, 2013.

[71] Fei Yang, Jue Wang, Eli Shechtman, Lubomir Bourdev, and Dimitri Metaxas. Expression flow for 3d-aware face component transfer. In *ACM Transactions on Graphics (TOG)*, volume 30, page 60. ACM, 2011.

[72] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. A 3d morphable eye region model for gaze estimation. In *European Conference on Computer Vision*, pages 297–313. Springer, 2016.

[73] Daniil Kononenko and Victor Lempitsky. Learning to look up: Realtime monocular gaze correction using machine learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4667–4675, 2015.

[74] Yaroslav Ganin, Daniil Kononenko, Diana Sungatullina, and Victor Lempitsky. Deepwarp: Photorealistic image resynthesis for gaze manipulation. In *European Conference on Computer Vision*, pages 311–326. Springer, 2016.

[75] Daniil Kononenko, Yaroslav Ganin, Diana Sungatullina, and Victor Lempitsky. Photorealistic monocular gaze redirection using machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[76] Sean Ryan Fanello, Cem Keskin, Pushmeet Kohli, Shahram Izadi, Jamie Shotton, Antonio Criminisi, Ugo Pattacini, and Tim Paek. Filter forests for learning data-dependent convolutional kernels. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1709–1716, 2014.

[77] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014. URL http://arxiv.org/abs/1412.6550.

[78] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[79] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2017. URL http://arxiv.org/abs/1503.02531.

[80] Reald. URL https://www.reald.com/#/home. Online; accessed 23-Aug-2017.

[81] George Wolberg. *Digital image warping*, volume 10662. IEEE computer society press Los Alamitos, CA, 1990.

[82] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[83] Brian A Smith, Qi Yin, Steven K Feiner, and Shree K Nayar. Gaze locking: passive eye contact detection for human-object interaction. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 271–280. ACM, 2013.

[84] Tadas Baltru, Peter Robinson, Louis-Philippe Morency, et al. Openface: an open source facial behavior analysis toolkit. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10. IEEE, 2016.

[85] Tadas Baltrusaitis, Peter Robinson, and Louis-Philippe Morency. Constrained local neural fields for robust facial landmark detection in the wild. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 354–361, 2013.

[86] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. Face alignment at 3000 fps via regressing local binary features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1685–1692, 2014.

[87] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

[88] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[89] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[90] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1841–1848, 2013.

[91] Antonio Criminisi and Jamie Shotton. *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media, 2013.

[92] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 775–781, 2005.

[93] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc J. Van Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision*, 101(3):437–458, 2013.

[94] Pei Yin, Antonio Criminisi, John M. Winn, and Irfan A. Essa. Tree-based classifiers for bilayer video segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[95] Juergen Gall and Victor S. Lempitsky. Class-specific hough forests for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1022–1029, 2009.

[96] Donald G Bailey. Sub-pixel estimation of local extrema. In *Proceeding of Image and Vision Computing New Zealand*, pages 414–419, 2003.

[97] Daniil Kononenko. Project webpage machine learning-based gaze correction. URL http://sites.skoltech.ru/compvision/projects/gaze-correction. `http://tinyurl.com/gazecorr`, Online; accessed 17-Aug-2017.

[98] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[99] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[100] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

[101] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, pages 143–155, 1989.

[102] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456, 2015.

[103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[104] Daniil Kononenko. Gaze correction for videoconferencing. https://youtu.be/sw31vBxQUNs, 2014. [Online; accessed 17-Aug-2017].

[105] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[106] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[107] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.