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

List of Figures

List of Tables

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 [Glassner, 2014]. 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, [Shu et al., 2016] uses physically-based face rendering and [Wood et al., 2017] uses 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 [Upchurch et al., 2017].

FIGURE 1.2: Video interpolation on the KITTI dataset. The middle frame is predicted from left and right frames. Figure taken from [Liu et al., 2017].

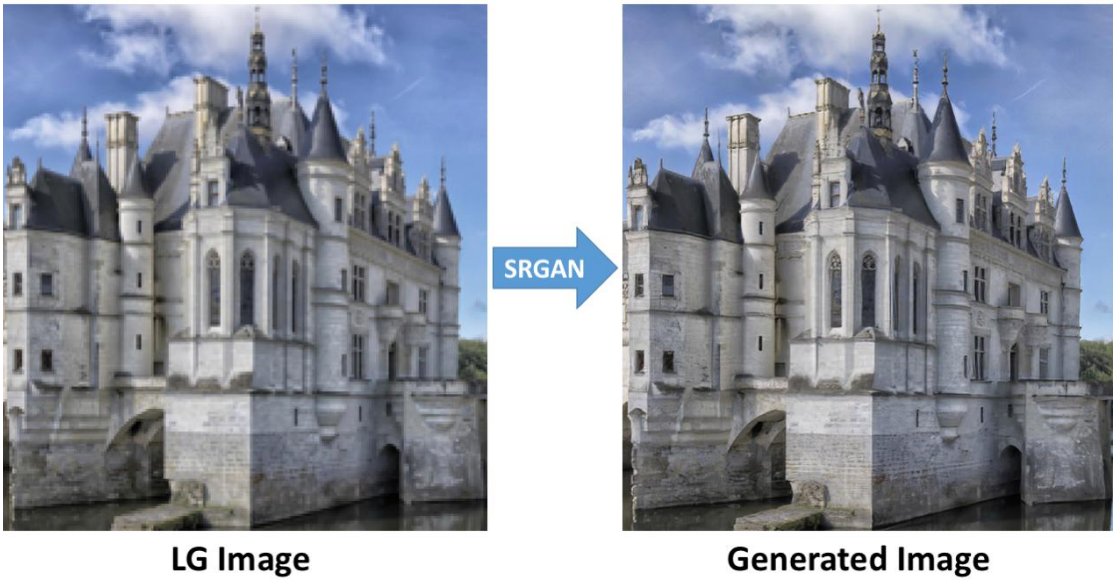


FIGURE 1.3: Image super-resolution. The resolution of the left image is enhanced on the fight one. The method utilizes the Generative Adversarial Networks approach (Section 1.1.1). Figure taken from [Ledig et al., 2017].

FIGURE 1.4: Examples of re-synthesis problem.

non-smiling faces from the CelebA dataset [Liu et al., 2015]. In some situations, aligned sets of examples of a different type can be obtained if ground-truth images are provided. In [Liu et al., 2017], for example, two input frames and the middle ground-truth image are picked from the UCF101 dataset [Soomro et al., 2012]. 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 [Mahendran and Vedaldi, 2015, Dosovitskiy et al., 2015, Goodfellow et al., 2014, Denton et al., 2015, Gatys et al., 2015, Gregor et al., 2015, Xue et al., 2016, Dosovitskiy and Brox, 2016]. 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 [Kulkarni et al., 2015, Reed et al., 2015] 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 [Zeiler and Fergus, 2014, Dosovitskiy et al., 2015], adversarial loss [Goodfellow et al., 2014], variational autoencoders [Kingma and Welling, 2013], perceptual loss [Dosovitskiy and Brox, 2016]). The recent paper [Xue et al., 2016] 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

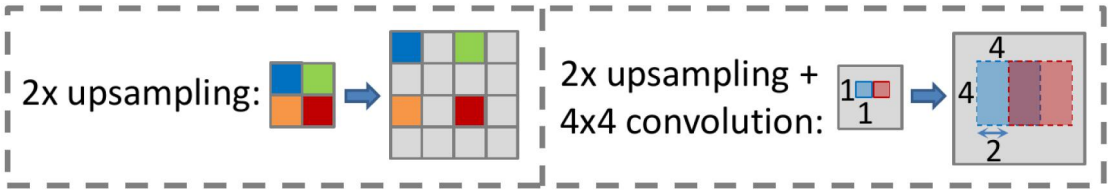


FIGURE 1.5: Illustration of upsampling (left) and upsampling+convolution (right) as used in the generative network. Figure taken from [Dosovitskiy et al., 2015].

warping field on the gaze redirection task. Before introducing further details on the proposed approach, I discuss the related works in recent literature.

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. [Dosovitskiy et al., 2015], 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 [Kingma and Welling, 2013, Rezende et al., 2014] (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

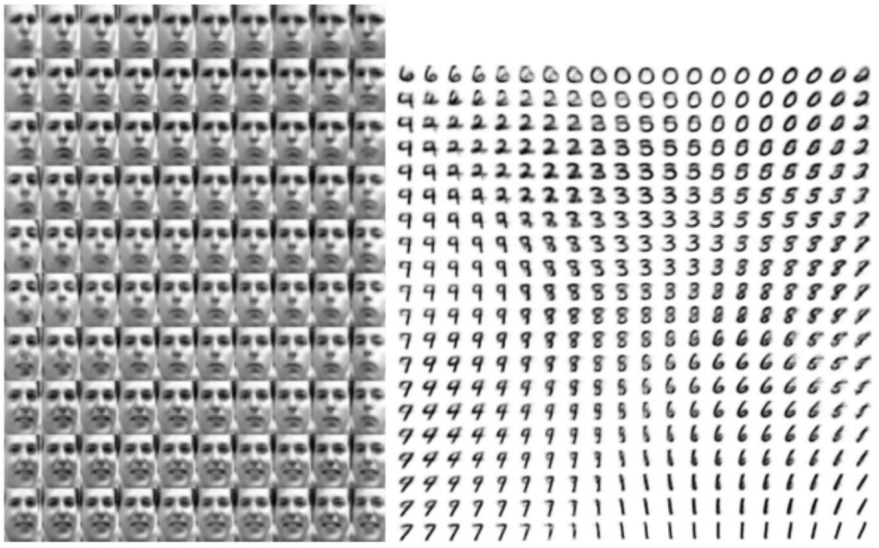


FIGURE 1.6: Examples of generated images using VAEs with 2D codes. Figure taken from [Kingma and Welling, 2013].

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.

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 [Kullback and Leibler, 1951]. 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 [Goodfellow et al., 2016]. 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 [Goodfellow et al., 2014] 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 [Zhu et al., 2016] and photo modifications [Brock et al., 2017, Isola et al., 2017] are based on GAN idea.

Interactive generative adversarial networks (iGAN) were developed in [Zhu et al., 2016]. 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, [Zhu et al., 2016] suggest to first project it on the approximation of the image manifold using the perception loss [Johnson et al., 2016] between intermediate deep neural network activations as a measure of closeness between the

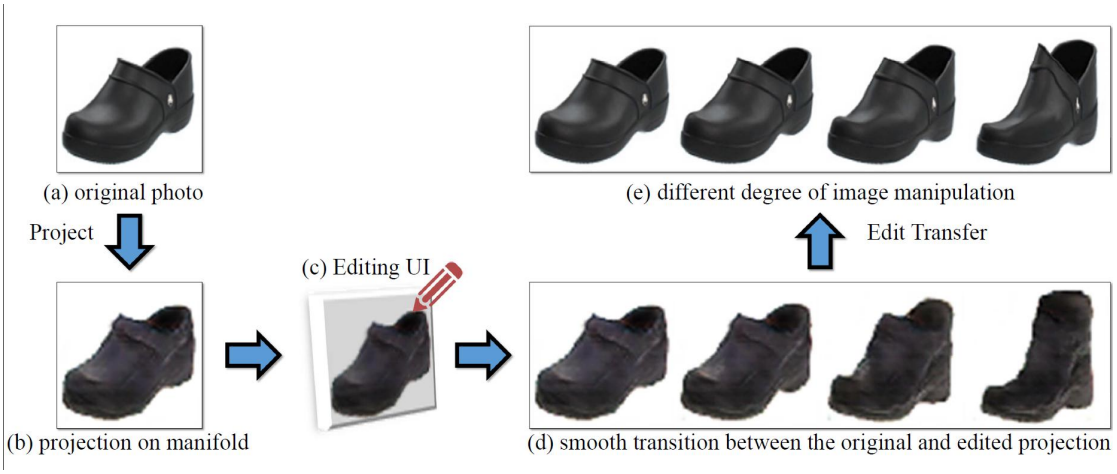


FIGURE 1.7: Image editing on the natural image manifold. The original photo is projected onto a low-dimensional latent vector representation, the color and shape of the generated image are modified, and the same modifications are applied to the original image. Figure taken from [Zhu et al., 2016].

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 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 [Isola et al., 2017]. 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 [Mirza and Osindero, 2014], 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 [Li and Wand, 2016, Isola et al., 2017] also suggest using

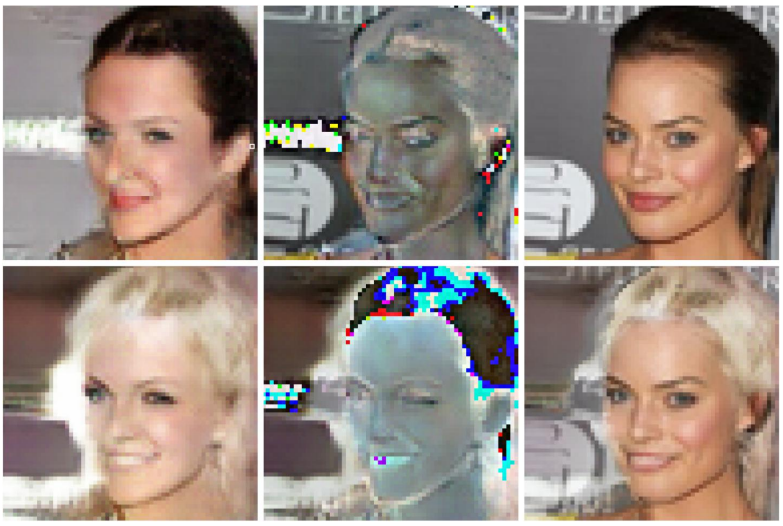


FIGURE 1.8: Photo editing in Neural Photo Editor. Top, left to right: Reconstruction, reconstruction error, original image. Bottom: Modified reconstruction, difference between modified reconstruction and $\hat{X}$, output. Figure taken from [Brock et al., 2017].

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.

An interface called Neural Photo Editor, where the user is able to manipulate the models latent variables in an interpretable way, was created in [Brock et al., 2017]. 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 [Brock et al., 2017] 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 [Dosovitskiy and Brox, 2016]. 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 [Krizhevsky et al., 2012] network trained on the image classification task.

[Shu et al., 2017] concentrate 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 [Land and McCann, 1971]) prior on the albedo and an assumption of low-frequency spherical harmonics-based lighting model [Ramamoorthi and Hanrahan, 2001, Basri and Jacobs, 2003]. 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 [Goodfellow et al., 2014] 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 [Seitz and Dyer, 1997]. 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

FIGURE 1.9: Examples of reconstructions produced by an encoder-decoder architecture that combines reconstruction ℓ_2 and GAN losses (following the approach in [Kulkarni et al., 2015, Ghodrati et al., 2016, Reed et al., 2015, Goodfellow et al., 2014]) 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 [Kulkarni et al., 2015, Ghodrati et al., 2016, Reed et al., 2015, Goodfellow et al., 2014].

discussions. Several papers present a related approach with a deep network predicting warping fields (flow) for image editing.

[Zhou et al., 2016] 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 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. [Zhou et al., 2016] 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

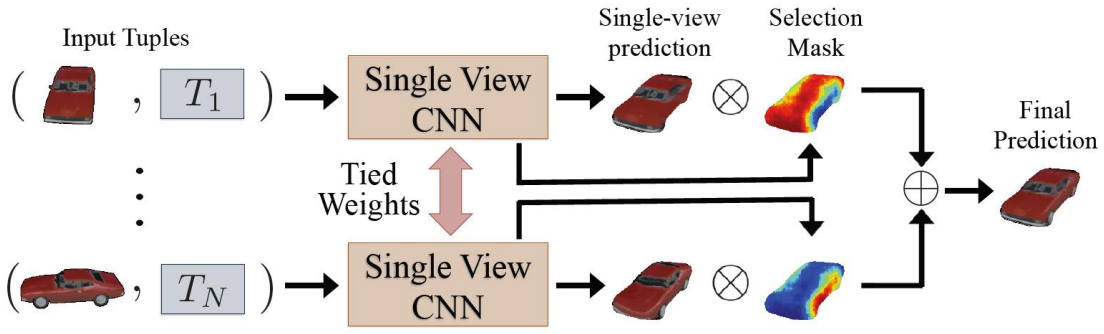


FIGURE 1.10: Multi-view network architecture ($\otimes$: per-pixel product, $\oplus$: per-pixel normalized sum). Shared-weights CNN generates a novel view and confidence mask for each input image. The output is the average of the novel views weighted with masks' values. Figure taken from [Zhou et al., 2016].

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.

[Yeh et al., 2016] 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 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 [Liu et al., 2017]. 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.

[Liu et al., 2017] also put 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.

The warping flow is used to fill in disoccluded pixels based on the prediction of the visibility map in [Park et al., 2017]. 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 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).

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,

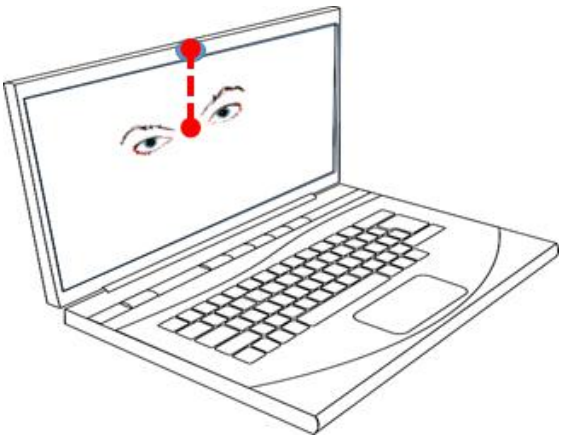
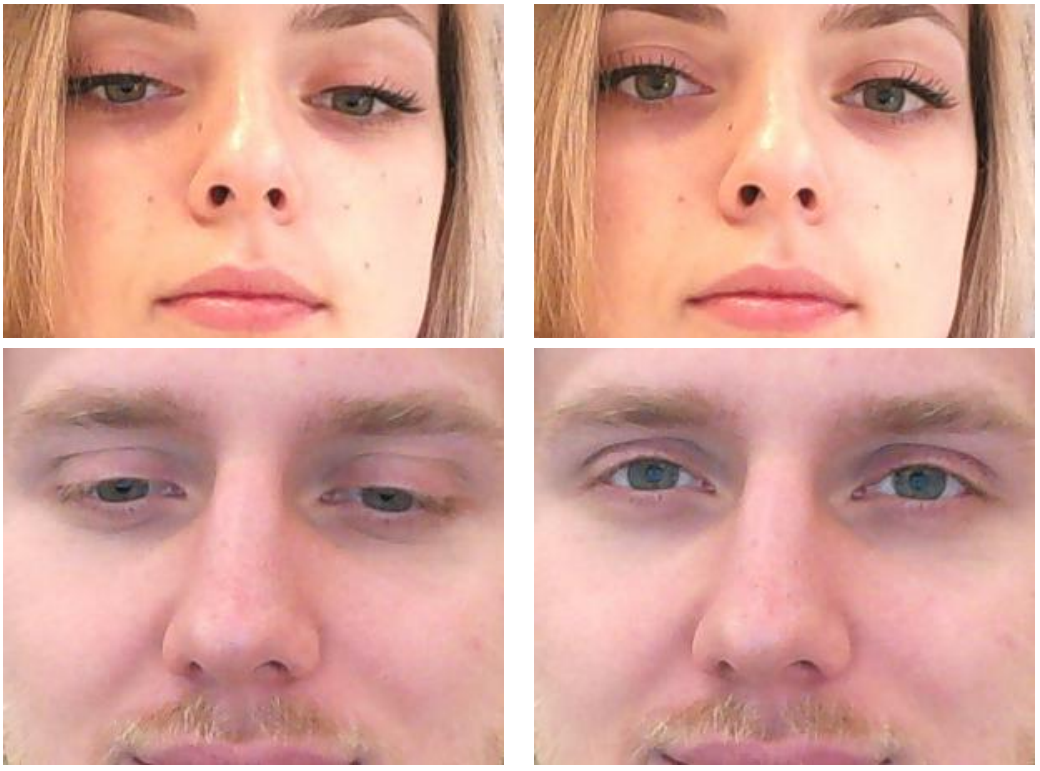


FIGURE 1.11: Eye-to-eye contact cannot be maintained during a videoconference due to a vertical gap between the camera and conference window on the screen. As a result, the person on the screen seems to be looking down while in fact they are looking at your face on their screen.



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.

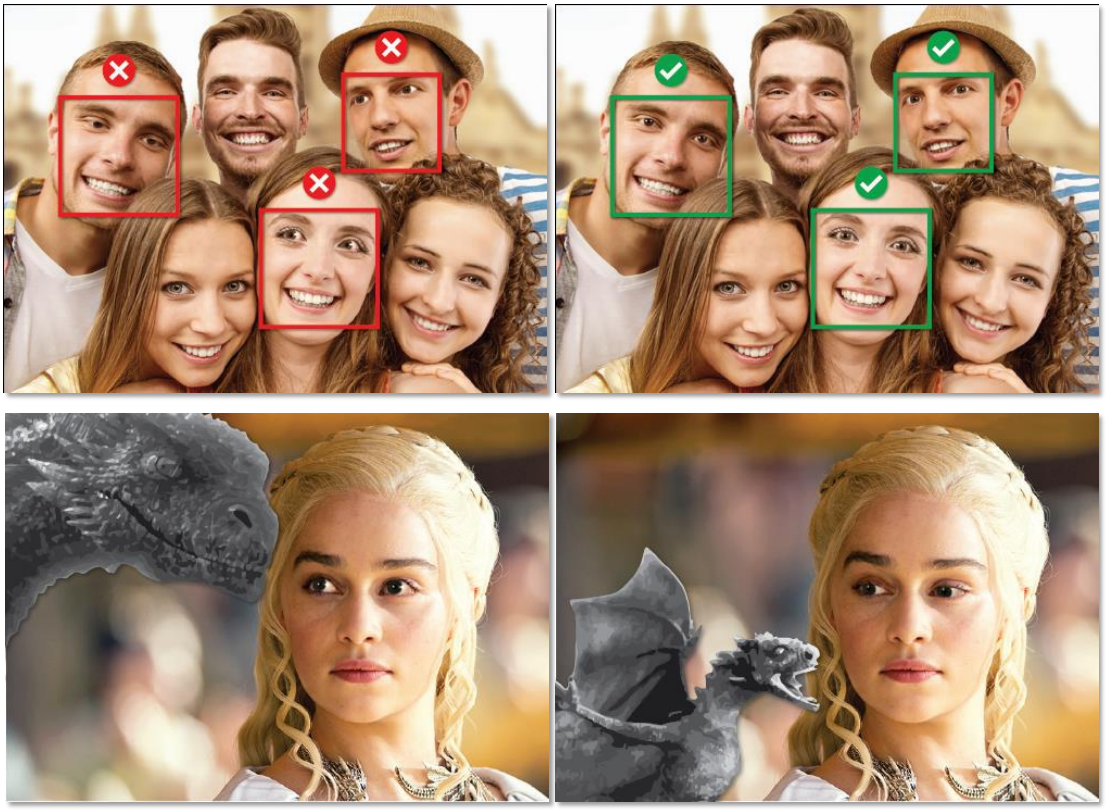


FIGURE 1.13: Gaze redirection for image and video post-processing. Top row: gaze correcting camera for the group of people. Bottom row: post-production of a movie after adding visual effects. CGI character is changed and the actor's gaze is adjusted accordingly. The method is based on constructing a complicated physical model of the eye. Figure taken from [Wood et al., 2017].

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 [Kleinke, 1986] on eye contact: "Authors of [Wheeler et al., 1979] 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 [Lochman and Allen, 1981]. 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 [Cook and SMITH, 1975]." 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 [Wallis et al., 2015]) 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 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 [Kleinke, 1986]. Thus meeting the requirement of realism is particularly difficult due to the well-known *uncanny valley* effect [Wikipedia, ndb], 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 [Okada et al., 1994, Jones et al., 2009], stereo cameras [Yang and Zhang, 2002, Criminisi et al., 2003], or RGB-D cameras [Zhu et al., 2011, Kuster et al., 2012]. 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 [Okada et al.,

1994, Jones et al., 2009]. The most popular software-driven approach to gaze correction in videoconferencing amounts to synthesizing 3D rotated views of either the entire scene [Criminisi et al., 2003, Yang and Zhang, 2002] or the face (which is subsequently blended into the unrotated head) [Kuster et al., 2012, Giger et al., 2014]. 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 [Yang and Zhang, 2002, Criminisi et al., 2003] or using RGB-D cameras [Zhu et al., 2011, Kuster et al., 2012]. 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 [Cham et al., 2002, Yip and Jin, 2003], 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 [Wolf et al., 2010, Qin et al., 2015] 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, [Shu et al., 2016] suggest a similar approach to replace closed eyes with open eyes for photo editing. [Wood et al., 2017] use a 3D eye model to generate realistic eye images with adjusted gaze.

The 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 [Jones et al., 2009] 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 [Yang and Zhang, 2002]. The method involves pose tracking, view matching and view

Paper	Monocular	Does not require GPU	Real-time	Does not require pre-calibration
[Kuster et al., 2012]	-	-	+	+
[Giger et al., 2014]	+	-	+	-
[Wolf et al., 2010]	+	+	+	-
[Criminisi et al., 2003]	-	+	∼+	+
[Yang and Zhang, 2002]	-	+	-	-
[Cham et al., 2002]	+	+	-	+
[Yip and Jin, 2003]	+	+	-	-
[Zhu et al., 2011]	-	+	-	+
[Qin et al., 2015]	+	-	+	-
[Wood et al., 2017]	+	-	-	+

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

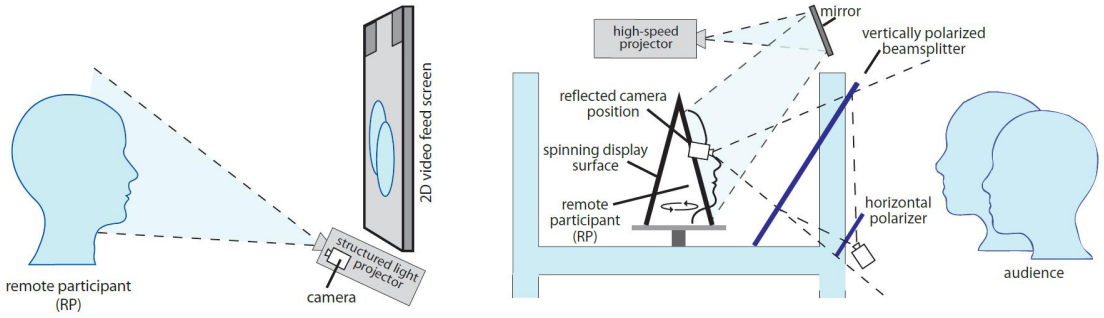


FIGURE 1.14: (Left) a setup for remote participant: the structured light scanning system (120Hz video projector and camera) and large 2D video feed screen. (Right) a setup for audience: the two-sided display surface, high-speed video projector, frontal beam splitter, and 2D video and face tracking camera. Crossed polarizers prevent the video feed camera from seeing past the beam splitter. Figure taken from [Jones et al., 2009]

.

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 [Liu et al., 2001] 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 [Hartley and Zisserman, 2003]:

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

where $\mathbf{F}$ is the fundamental matrix, corresponding to the epipolar geometry between the two images. The points are tracked using the KLT tracker [Shi and Tomasi, 1994].

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 [Seitz and Dyer, 1997] or on hardware-assisted multitexture blending.

[Criminisi et al., 2003] suggest 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 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.

[Zhu et al., 2011] suggest 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 [Zhu et al., 2008, Zhu et al., 2009]. View morphing [Seitz and Dyer, 1997] in case of a dense correspondence is performed through linear interpolation of projection matrices. A texture splatting technique [Guinnip et al., 2004] is used to fill the holes emerging from matching, projection and rounding errors.

[Kuster et al., 2012] suggest generating 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

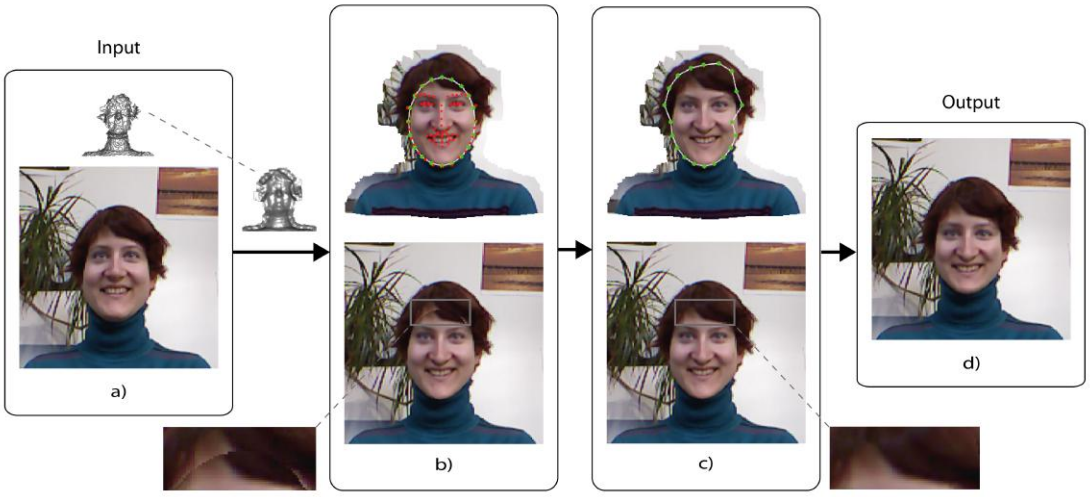


FIGURE 1.15: a) Input color and depth images. b) Top: synthesized image. Bottom: transfer with the use of an ellipse stencil; visible artifacts. c) Seam is optimized; much fewer artifacts. d) Final blending. Figure taken from [Kuster et al., 2012]

.

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 [Kuster et al., 2012] is the necessity to use Kinect sensor and GPU to compute fast enough for real-time performance.

This approach is further developed in [Giger et al., 2014] 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 [Saragih et al., 2011]. Conversion between the 2D landmark and 3D world coordinate systems is performed using the Laplacian deformation technique [Sorkine, 2005]. 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 [Giger et al., 2014] has some limitations, due

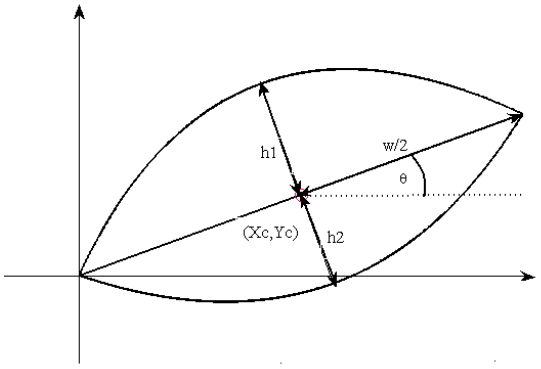


FIGURE 1.16: Eye model consisting of 6 parameters. Figure taken from [Wolf et al., 2010]

.

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 [Cham et al., 2002] 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. [Yip and Jin, 2003] suggest 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 [Wolf et al., 2010] 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 [Yuille et al., 1992].

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 [Everingham et al., 2006] 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).

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 [Wolf et al., 2010] 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 [Qin et al., 2015] 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 [Wolf et al., 2010]: the gaze is unnaturally directed at the camera all the time and each videoconferencing session requires a set of additional preparatory operations. Face alignment [Xiong and De la Torre, 2013] 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, [Shu et al., 2016] suggests an approach similar to that addressed in [Wolf et al., 2010] 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 [Saragih et al., 2011]. Assuming a weak perspective projection, they optimize for a projection matrix and weights in the PCA decomposition using the fitting algorithm [Yang et al., 2011]. 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, [Wood et al., 2017] was published after the experimental part of this study had already been complete. [Wood et al., 2017] model the eye region in 3D, recovering the shape and pose of the eyes [Wood et al., 2016], 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 [Wood et al., 2017] 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, 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 [Kononenko and Lempitsky, 2015] 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

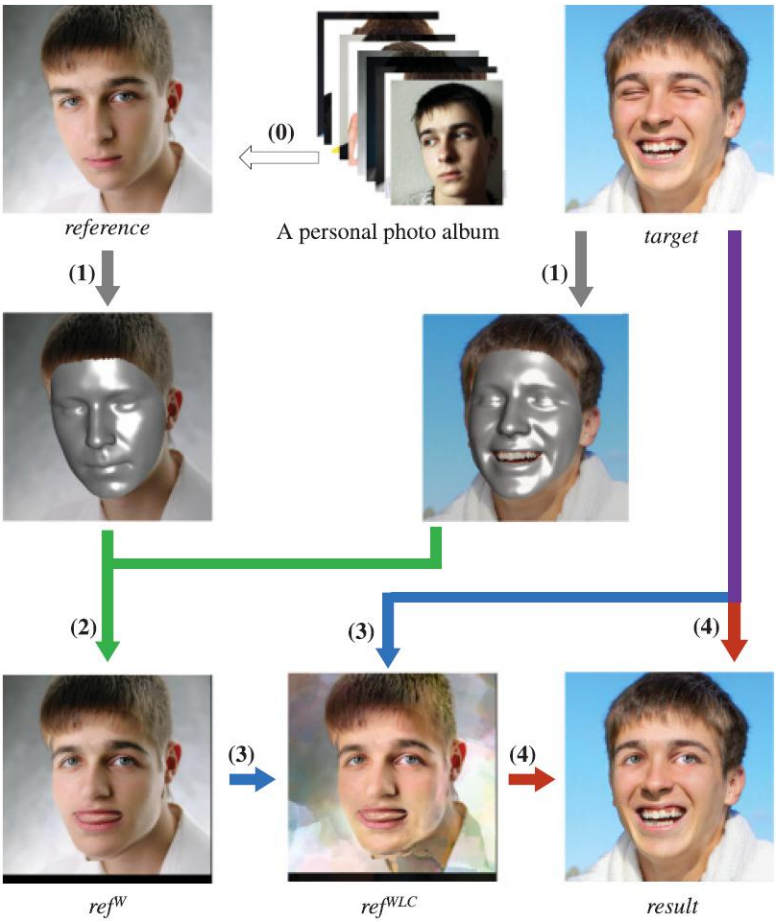


FIGURE 1.17: Eye-editing approach [Shu et al., 2016]. Steps applied to transfer the eyes: face fitting, eyes' registration via warping, color adjustment, blending. Figure taken from [Shu et al., 2016].

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, [Ganin et al., 2016]) 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, [Kononenko et al., 2017]) 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 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

Method	Inputs	Training data	Computational demands	Quality of results
Weakly-supervised Random Forest (Chapter 3, [Kononenko and Lempitsky, 2015])	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, [Ganin et al., 2016])	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), [Kononenko et al., 2017]	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.

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 [Seitz and Dyer, 1997]. However, the warping field in [Seitz and Dyer, 1997] 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.

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. [Fanello et al., 2014] predict 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 [Romero et al., 2015], or an ensemble of networks [Bucilu et al., 2006, Hinton et al., 2015]. 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.

[Reed et al., 2015] suggest 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 [RealD, nd].

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 discussed 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. The terms "warping field" and "flow field" are sometimes used interchangeably in this manuscript.

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 the input image pixels to emulate transformation. I rely on machine learning to accomplish this. Direct regression of the output image pixels has problems with synthesizing high-resolution images ([Kulkarni et al., 2015, Ghodrati et al., 2016, Reed et al., 2015, Goodfellow et al., 2014], Figure 1.9, Figure 2.2, Figure 2.3)), so in this work I suggest using the warping-based approach instead. 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 the case of a 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 can be easily generalized to a case where 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 bilinear interpolation [Wolberg, 1990]. The intensity value is interpolated depending on the distance to the four closest points in the pixel grid (Figure 2.1). If we specify $\tilde{x} = x + u(x, y)$, $\tilde{y} = y + v(x, y)$, $x_1 = \lfloor \tilde{x} \rfloor$, $x_2 = \lceil \tilde{x} \rceil$, $y_1 = \tilde{y} \rfloor$, $y_2 = \tilde{y} \rceil$, 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 a piece-wise differentiable [Jaderberg et al., 2015] and thus it can be incorporated into a deep architecture with end-to-end training based on gradient descent. This is further discussed in Chapter 4 and Chapter 6.

The methods presented in the following chapters are trained at a fixed spatial scale. At test-time, some of them can be applied at the native resolution of the input image (forest-based methods), while others 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.

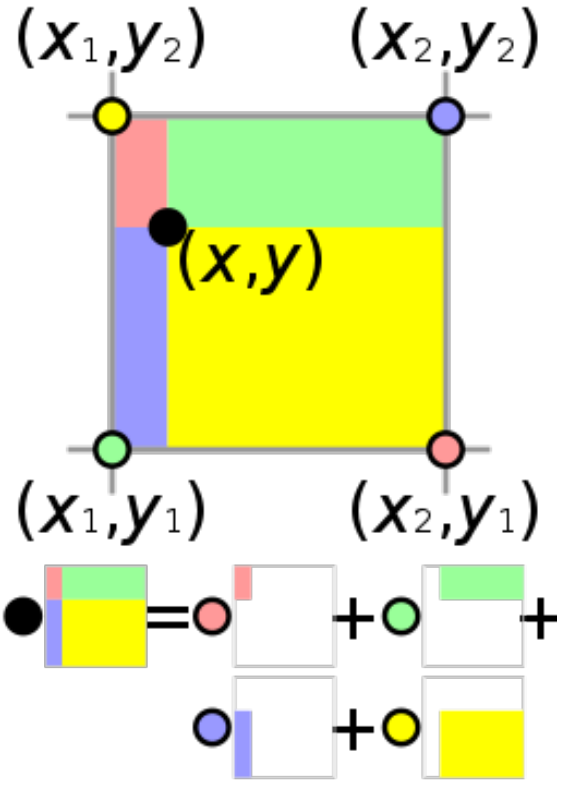


FIGURE 2.1: Visualization of bilinear interpolation. The value at the black spot is the sum of the value at each colored spot multiplied by the area of the rectangle of the same color, divided by the total area of all the four rectangles. Image taken from [Wikipedia, nda].

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

A warping "copy-pasting" procedure (2.2) is a restriction on the transformation family. It introduces natural regularization in the suggested method. This approach ensures that the output image pixels are copied from the input rather than invented. I give an example comparing the warping approach to direct regression on a gaze redirection task in Figure 2.2 and Figure 2.3. While the unregularized model suffers from noticeable fine-scale detail omissions and regression-to-mean effect, the warping approach produces photorealistic results. Moreover, the warping model shows comparable results for the training and testing sets, while the non-warping model suffers from overfitting. This chapter offers a summary of the model. For the full description of the model and training process used in the comparison, see Chapter 6.

Both warping and non-warping models are trained on pairs of eyes of the same person with a differing unknown gaze direction. Both eyes are encoded into low-dimensional

FIGURE 2.2: Warping and non-warping approaches compared; training set. Columns from left to right: input image, ground-truth image, warping model result, non-warping model result. Significant effects, such as omission of fine details, noise and regression-to-mean are decreased using the warping-based approach (2.2)

.

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

In some cases, however, 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 the 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 an appropriate learning method should be developed to handle such weak supervision. Examples of warping fields for gaze redirection are visualized the Figure 5.1.

FIGURE 2.3: Warping and non-warping approaches compared; validation set. Columns from left to right: input image, ground-truth image, warping model result, non-warping model result. The non-warping model exhibits significant overfitting as compared to train results Figure 2.2.

.

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 respective predictor learning methods proposed in Chapter 3, Chapter 4, Chapter 5, Chapter 6, are the key point of discussion within this study.

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

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

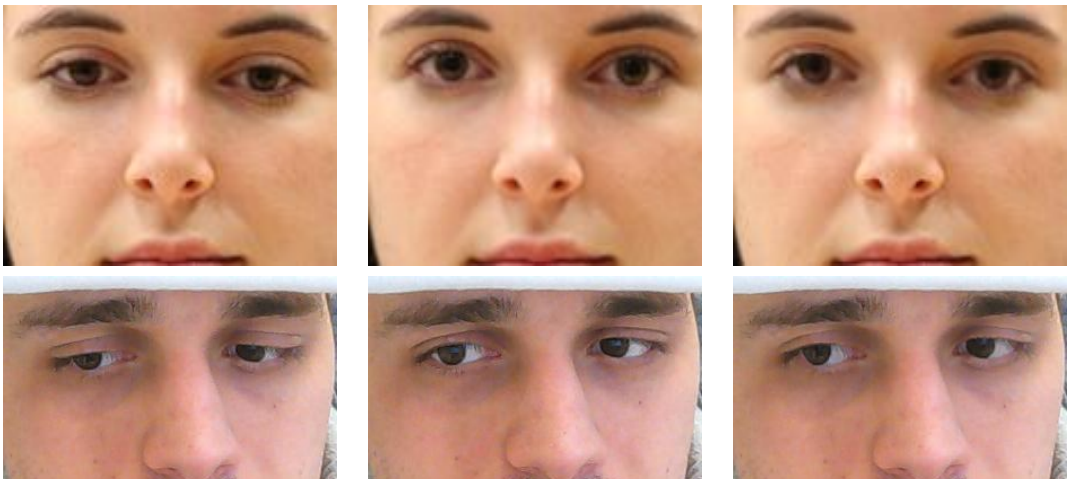
In the rest of my thesis, I concentrate on the gaze redirection task, however, all the suggested predictor learning methods are applicable to this general image re-synthesis pipeline. The restriction on the image re-synthesis problem where the warping is applicable, is that pixels of the output image should be 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 of the previous gaze correction systems synthesize a novel view of a scene from a virtual viewpoint co-located with the screen [Yang and Zhang, 2002, Criminisi et al., 2003, Cham et al., 2002, Yip and Jin, 2003]. Alternatively, a virtual view restricted to the face region is synthesized and stitched into the original video stream [Kuster et al., 2012, Giger et al., 2014]. 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 multi-perspective 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,



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 an example of the system proposed in Chapter 3 as applied to an image from the Columbia Gaze dataset [Smith et al., 2013]. The bottom row is an example of the system from Chapter 5 as applied to an image from the Skoltech dataset Section 2.3. In this particular example, the methods computation time is 8 ms for the top row and 5 ms for the bottom row on a single laptop core (excluding feature point localization). Such speed makes both systems suitable for real-time use in videoconferencing.

- complex occlusion/dis-occlusion of the eyeballs by the eyelids,
- change in illumination patterns due to 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. Another apparent plus of gaze redirection as compared to view synthesis is that it can be performed locally in the vicinity of each eye and thus will affect a small proportion of pixels in the image. At runtime, all the suggested methods localize each eye and then perform local operations with eye region pixels to accomplish gaze redirection.

2.3 Database collection and eye localization

The publicly available Columbia Gaze dataset [Smith et al., 2013] for the 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: combinations of seven horizontal ones (0°, ±5°, ±10°, ±15°) and three vertical ones (0°,

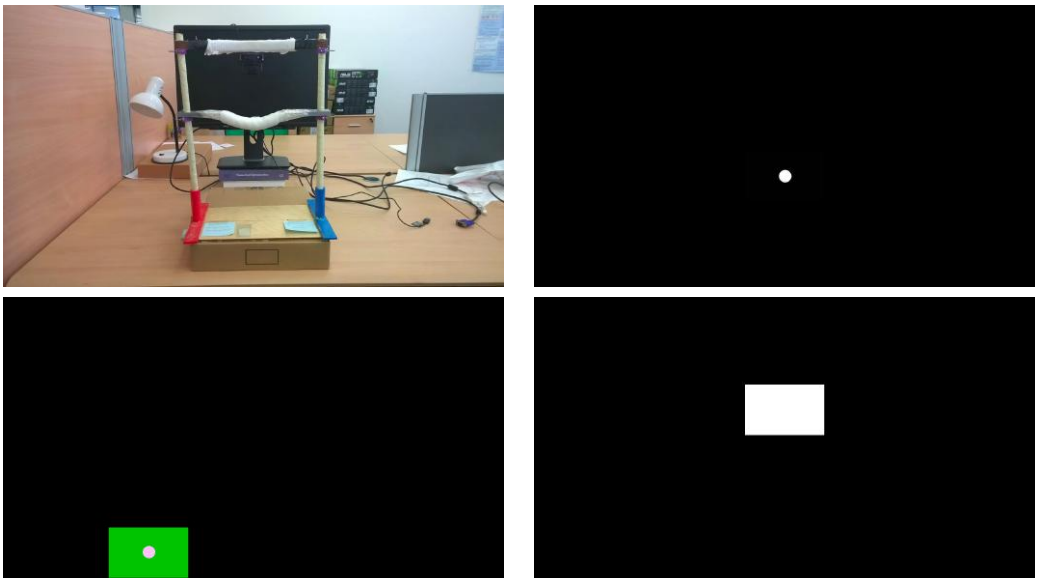


FIGURE 2.5: Dataset collection process. A person is asked to follow the point on the screen with the eyes only, without moving their head. A special head stand is used to minimize possible head shaking. (Top left): the setting, (top right): the point is moving, (bottom left): the point is stationary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

$\pm 10^\circ$). Taking the pairs with the same parameters except for the vertical gaze direction, I draw training samples for learning to redirect the gaze by 10 degrees.

However, there are several limitations associated with the Columbia Gaze dataset. The first is a poor variety of vertical redirection angles. The main testbed for gaze correction in videoconferencing is 15° upward redirection (Figure 1.12) – a case for which the Columbia dataset would hardly provide any examples. Second, images in this dataset are only loosely registered, which introduces an additional error and prompts the need for more registration procedures (Section 3.2.4). And finally, there are not enough training examples to learn a fully generalizable model.

The Skoltech Dataset of videos of around 150 people was put together specifically to overcome these restrictions. The recording was made so as to minimize head movement: the person was asked to place their head on a special stand and follow the moving point on the screen in front of the stand with their gaze. The sequence of frames is synchronized with the point position, from which we can deduce the gaze direction, was recorded using a webcam mounted in the middle of the screen (Figure 2.5). Some examples taken from one sequence in the dataset are presented in Figure 2.6.

About 200 frames are recorded for one video sequence. The angular range is 36° in vertical direction and 60° in horizontal direction. Bad shots, where a person is blinking,

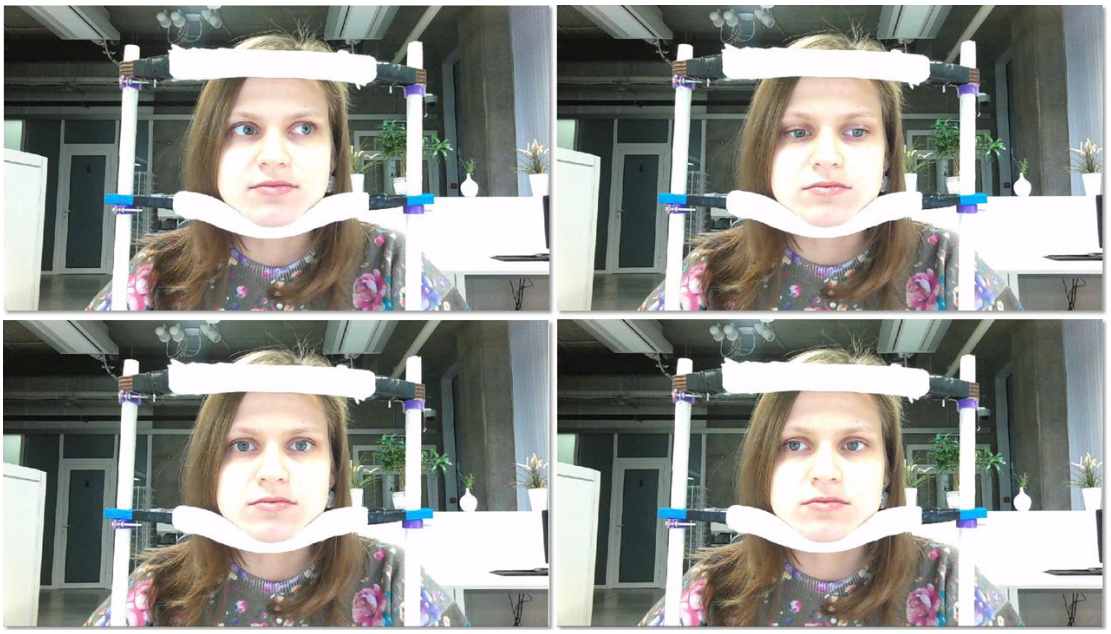


FIGURE 2.6: Examples from the dataset. The data collected consists of images of a person changing gaze direction only with their eyes, without moving their head.

not changing gaze direction monotonically as anticipated or moving their head are discarded manually. 2 to 10 sequences with different head poses and lighting conditions are recorded for each person (a total of 500 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 the systems proposed in this work start by localizing the eye regions and then do redirection by localized processing. The training samples are cropped using the eye localization procedure. The eye localization step within my system is a standard procedure, for I use an off-the-shelf real-time face alignment library (e.g. [Xiong and De la Torre, 2013, Baltru et al., 2016, Baltrusaitis et al., 2013]) 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 eyes.

For each eye, I focus on the landmark points $\mathbf{l}_1 = (l_1^x, l_1^y), \mathbf{l}_2 = (l_2^x, l_2^y) \ldots \mathbf{l}_N = (l_N^x, l_N^y)$ corresponding to that eye (in the case of [Baltru et al., 2016] there are $N = 7$ feature points). I compute a tight axis-aligned bounding box $\mathcal{B}'$ of those points. Next, the final bounding box $\mathcal{B}$ is defined 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.

Two variants of defining the characteristic radius Δ are proposed:

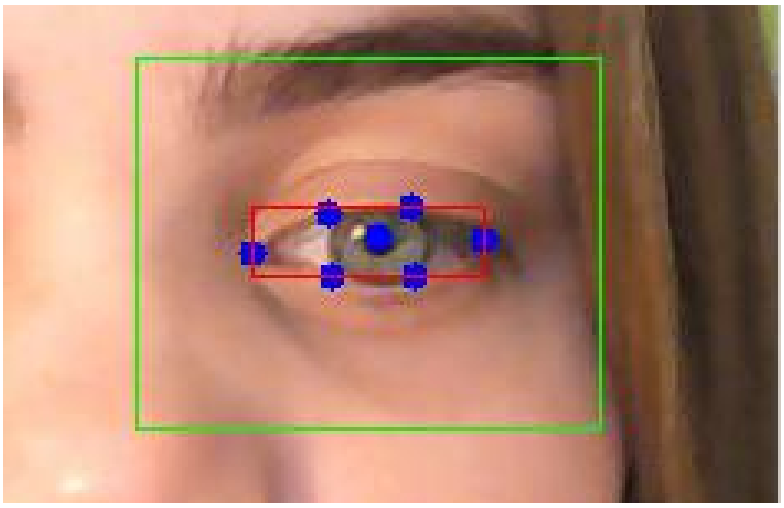


FIGURE 2.7: Illustration of eye localization (variant (2.8)). Blue: eye landmarks; red: tight bounding box around landmarks; green: bounding box enlarged proportionally to the distance between the corner landmarks that all modifications are restricted to in the suggested approach.

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

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

- the tight bounding box width equal to 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. Chapter 3 features the results for the first variant (2.7) and the following chapters provide the results for the second variant.

The left and right eyes are incorporated into one dataset, mirroring right eyes. At test time, the same predictor can be used for the 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.

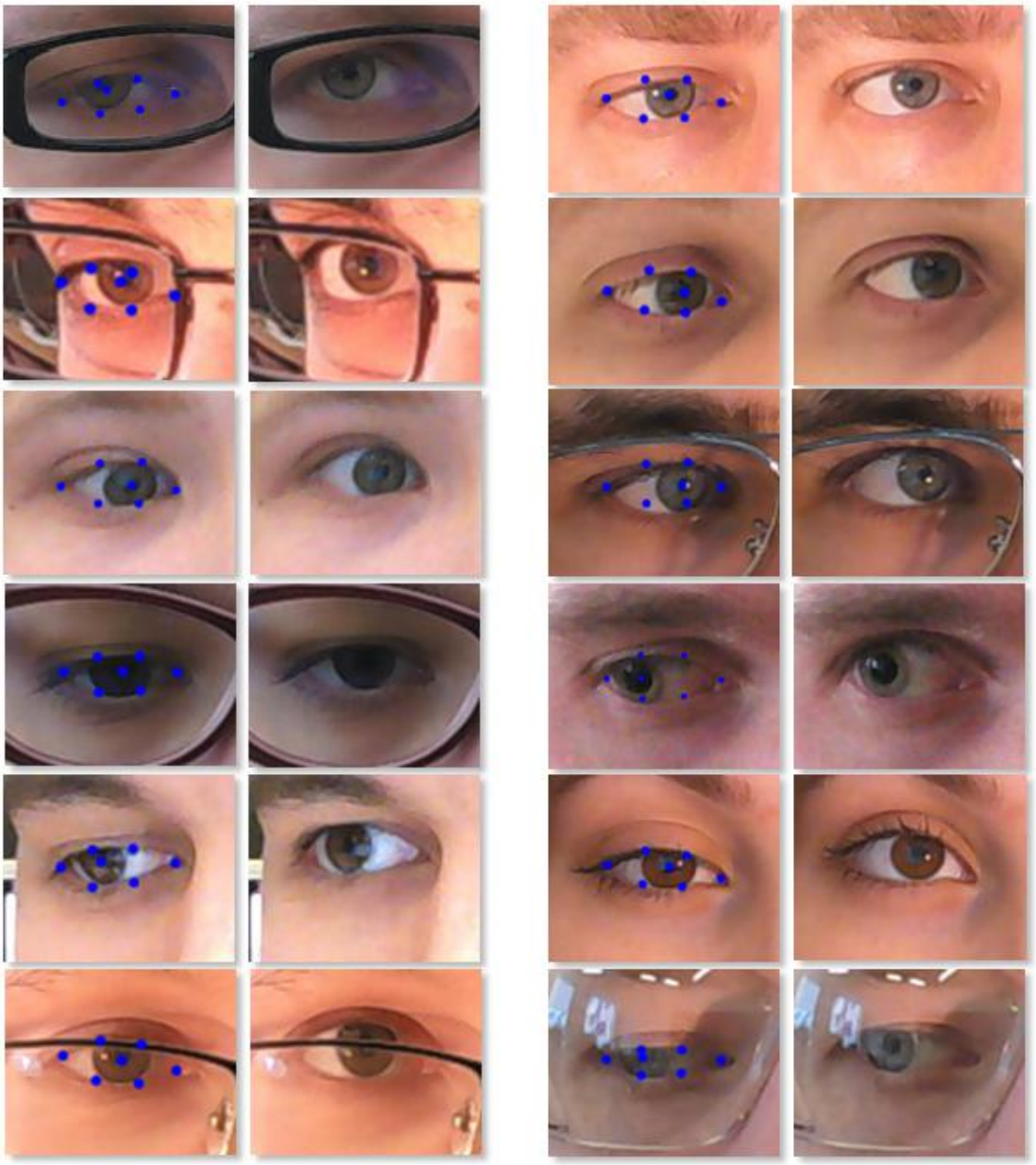


FIGURE 2.8: Examples of training data.

2.3.1 Related work on Facial Alignment

To crop eye images from training examples, I use facial alignment software [Xiong and De la Torre, 2013, Baltru et al., 2016, Baltrusaitis et al., 2013]. I also use tracked landmarks as additional features for a training model. This section gives an overview of related work on facial alignment methods.

[Xiong and De la Torre, 2013] consider facial alignment as a Nonlinear Least Squares problem and adapts Newton's method to solve it. Using the training data, the authors

learn a series of parameter updates, minimizing the objective. The updates are decomposed into person-specific components and generic descent directions. The method learns average descent directions during training. In testing, given an image of an unseen person, an update is generated by projecting person-specific components onto the learned generic directions.

[Kazemi and Sullivan, 2014] also treat the task as a regression problem and suggests tracking 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 the 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 and $r_t(I, \mathbf{S}$ is an update given by the regressor in the cascade. The initial shape is the mean landmark position along the training data, which is centered and scaled based on the bounding box output of a face detector. During training, the initial shape is picked up randomly from the training data for data augmentation purposes. In order to have shape-invariant split tests, during training the 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. A generalization to the case of missing labels is suggested. The approach works superior to real-time: the speed is up to 1000 FPS.

[Ren et al., 2014] also adapt the cascaded pose regression, with the local binary features learned with a random forest individually for each facial landmark. The local binary features thus obtained are used to jointly learn linear regression for the final output.

[Baltrusaitis et al., 2013] present the Constrained Local Neural Field model for facial landmark detection, proposing 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 is applied, which takes into account the reliabilities of each patch expert.

2.4 General system for gaze redirection

Although the suggested systems differ in many aspects, such as the training data, actual type of predictor used, etc., there are a number of standard steps which all the systems follow when processing a new face image. For the system-specific details (such as predictors), please refer to the comparison presented at 1.2 and the next chapters. This section

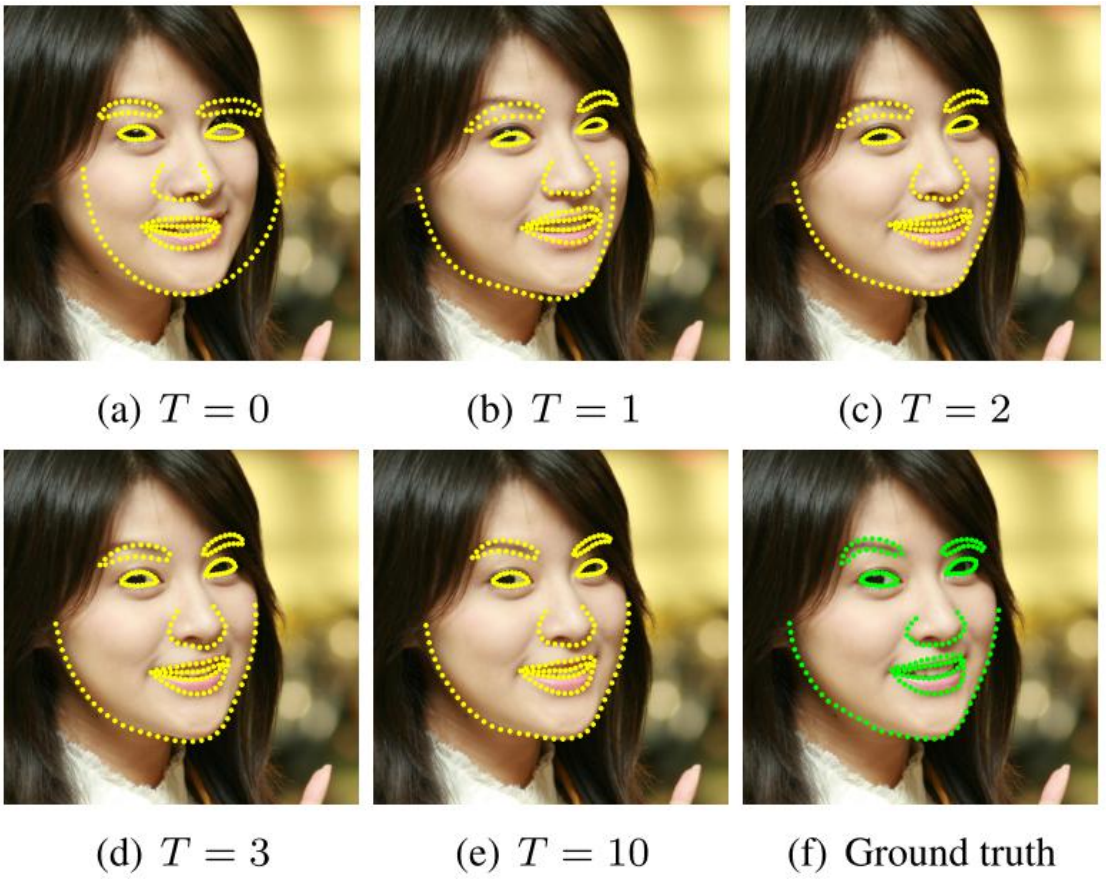


FIGURE 2.9: Landmark estimates at different levels of the cascade. Figure taken from [Kazemi and Sullivan, 2014].

deals with the general image re-synthesis pipeline (Section 2.1.1) specifically adapted to the gaze redirection task.

1. A facial alignment software is applied using a face image as input (currently [Baltrusaitis et al., 2013, Baltru et al., 2016] are used) and producing facial landmarks as output (see the example in Figure 2.9). Landmarks $\mathbf{l}_l$ and $\mathbf{l}_r$ are selected, corresponding to the left and right eyes, respectively. If there are more than one person in the image (and all the systems rely on the facial landmark software to detect all the faces), the next steps are applied successively to all the faces.

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

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

4. The warping field is predicted
$$\mathcal{F} = p(I, \mathbf{l}).$$
The procedure at hand is specific to each system, because different systems have different inputs (redirection angle or analogy pairs of images are possible additional inputs).

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

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

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

8. The right eye is mirrored back and both eyes are pasted back in to get the output image. Since the eye images are cropped from the input face images, in order to blend the result back without stitching the artifacts, the flow maps are linearly decreased to zero near the cropped eye border.

The procedure outlined above reflects the general approach without specifying how the warping field $\mathcal{F} = p(I, \mathbf{l})$ is predicted. The next chapters will describe several systems that detail and expand on the general approach presented 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 which requires collecting and processing a large number of image pairs describing the target transformation. A 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 the clear potential for real-time performance on tablet computers. Monocular gaze correction in videoconference is used as a test case to illustrate and evaluate the system performance. The system synthesizes realistic views with a gaze systematically redirected upwards (10 or 15 degrees in my experiments). At test time, the system provides gaze correction using simple pixel replacement operations that are localized to the vicinity of the persons eyes, thus achieving high computational efficiency.

3.1 Overview of Random Forests

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

The decision tree is a tree graph (graph without cycles) with the nodes divided into two groups: split nodes and leaf nodes. The split nodes are all the internal (and 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 and $\mathcal{T}$ is the parameter space. For each data point, the tree is traversed from root to leaf, so that at each split node, the test function result determines whether the data point proceeds to the left or right child. A common choice in random forest applications is decision stumps, i.e. the data split 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 an unsupervised scenario, a training point $\mathbf{s} = \mathbf{x}$. I also write down (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:

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

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

At test time, a new query sample 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 a split function (3.1), e.g. assigning some value to a set of parameters θ_j, is a key issue in applying decision trees. This 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 recursive greedy manner from root to leaves. Training begins at the root node, $j = 0$. A training set is divided into two disjoint subsets at 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 can be used, including a few most common ones:

- the maximum depth of the tree D has been reached;
- some minimum threshold value of the information has been gained (3.4), which means that the intended attributes of the training point are the same across all the samples in the current node, or at least, one does not gain enough information from dividing samples in the current node into two new sets;
- the 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, the 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 a set of all classes and $p(c)$ is empirical distribution over the classes in 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 a regression problem is the 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 the points in the node.

Once the learning phase in a supervised problem is complete, each leaf node l contains a subset of labeled training data. In a general case, the leaf statistics define a distribution

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

which, for the classification and regression cases, can be conditional distribution of the categorical label $p(c|\mathbf{x})$ or the 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 end up in a leaf containing similar training points. Thus a reasonable assumption for labeling this new test point would be to put a label similar to the training points in that leaf. For example, Maximium-A-Posteriori of the leaf label statistics can be used:

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

In the case of classification, this corresponds to picking a class with the highest frequency and in the case of regression the mean of all the training points falling within a particular leaf.

3.1.2 Random Forests

Introduced in [Amit and Geman, 1997, Breiman, 2001], Random Forest is an ensemble of several Decision Trees. There are two principal ways of injecting randomness:

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

Bagging was introduced in [Breiman, 1996, Breiman, 2001] as an overfitting reduction method. Each tree in the forest is trained on a random subset picked from the full training set. Such a method increases generalization, while preventing specialization of all the trees on a single dataset. This strategy will also make training faster for each tree.

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

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

Optimization is normally performed via brute force search over the set $\mathcal{T}_j$. For a typical case of decision stumps (3.2), one may wish to randomize either the feature number i or threshold τ, or both. For example, [Dollár and Zitnick, 2013] consider some small number of features for each split train. All the subsets of samples in the node are sorted according to the chosen feature value and thus the optimal threshold value is found linearly with the subsample size.

More detailed definitions of the abstract Decision Tree and Random Forest models can be found in [Criminisi and Shotton, 2013].

3.1.3 Applications 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 [Amit and Geman, 1997, Breiman, 2001], including real-time object pose estimation [Lepetit et al., 2005], head pose estimation [Fanelli et al., 2013], human body pose estimation [Shotton et al., 2013], background segmentation [Yin et al., 2007], etc.

[Lepetit et al., 2005] consider such problems as object detection and pose estimation. They match keypoints from the input image with the ones on the target object. Formulating the point matching as a classification problem, they use random forests to solve it. Formally, at training time they select a set $\mathcal{K}$ of cardinality K oof the keypoints in the object model. At test time they select a patch, centering it around the keypoint found. The task is to determine whether it matches one of K keypoints in $\mathcal{K}$ or not, i.e. to solve a classification problem with $K+1$ labels $\{-1, 1, 2, \ldots, K\}$, where -1 means the keypoint does not belong to $\mathcal{K}$. They suggest to use difference of pixels intensities from the neighborhood of the keypoint as a binary node tests (3.1) in classification trees. The authors compare two tree growing techniques. In one they pick several tests at random and choose the best according to the information gain (3.4), as was shown above. The number of tests picked is 10 at the root node and $100d$ for a node at depth d, $d \geq 2$. The other is referred to as extremely randomized trees and involves picking only one test at random, with no optimization applied. This approach helps significantly decrease the training time while keeping the quality loss at a low level.

[Shotton et al., 2013] put forward two human body pose estimation approaches: body part classification (BPC) and offset joint regression (OJR), both employing random forest and applying it individually to each pixel in a depth image. BPC predicts body part labels for each pixel in the image. A density over the 3D world space can be calculated by re-projecting per-pixel label probabilities, based on the known depth image and the known calibration of the depth camera. OJR instead directly casts 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 the features depth invariant. The authors choose approximately half of the tests in such a way that one of the two pixels compared 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. [Gall and Lempitsky, 2009, Dollár and Zitnick, 2013]).

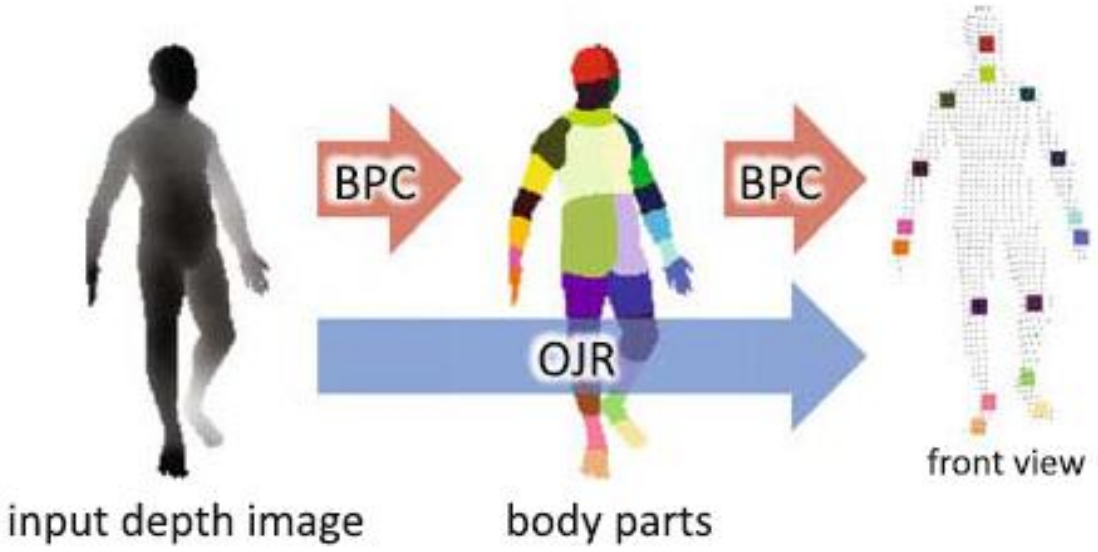


FIGURE 3.1: Body part classification (BPC) first predicts a (color-coded) body part label at each pixel and then uses these inferred labels to localize the body joints. Offset joint regression (OJR) instead directly regresses the joint positions. Figure taken from [Shotton et al., 2013]

.

A Hough-transform-based object detection method is suggested in [Gall and Lempitsky, 2009]. They suggest to train a forest to map the appearance of an image patch into its Hough vote, incorporating it in the decision forest framework. The 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 the 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 on whether to optimize the split based on one or another with equal probabilities.

[Dollár and Zitnick, 2013] apply structured random forests to edge detection. In their approach, $\mathbf{y} \in \mathcal{Y}$ is some structured image annotation, e.g. an edge mask. The idea is to map all the examples falling within 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)) can be used to train a split.

[Dollár and Zitnick, 2013] first use intermediate mapping $\Pi : \mathcal{Y} \to \mathcal{Z}$ to some Euclidean space $\mathcal{Z}$ and then employs some clusterization technique to obtain discrete labels $\mathcal{C} = \{1, \dots, k\}$. As applied 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 pixels lies on the same or different vectors. In practice, only a subset of $m = 256$ dimensions of these binary vectors is sampled, which is then reduced to 5 dimensions by PCA.

The idea of the ensemble model combining a set of n labels $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathcal{Y}$ into a single prediction is to use the mapping $\Pi : \mathcal{Y} \to \mathcal{Z}$ to the Euclidean space and then choose a medoid sample, i.e. among $z_k = \Pi(y_k)$ choose k such that z_k minimizes the sum of distances to all other z_i. The ensemble model is needed both to set the label in the leaf node and to merge predictions from several trees into a single one.

The trees in my method are trained under weak supervision, which makes my work related in a way to the methods from [Fanello et al., 2014]. The suggested filter forests learn 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 $\mathbf{y}_i \in \mathcal{Y}$ stands for a filtered patch, e.g. a denoised patch) does not contain target filters. [Fanello et al., 2014] 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. A similar approach to learning optimal warping flow vectors is applied in the method introduced in Section 3.2.3. However, in addition to this idea, not only the optimal filter, but the error distribution depending on filters is stored in the node. Furthermore, an unconventional way of ensembling the trees in the context of weak supervision is also suggested (Section 3.2.2).

3.2 Warping flow forest

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

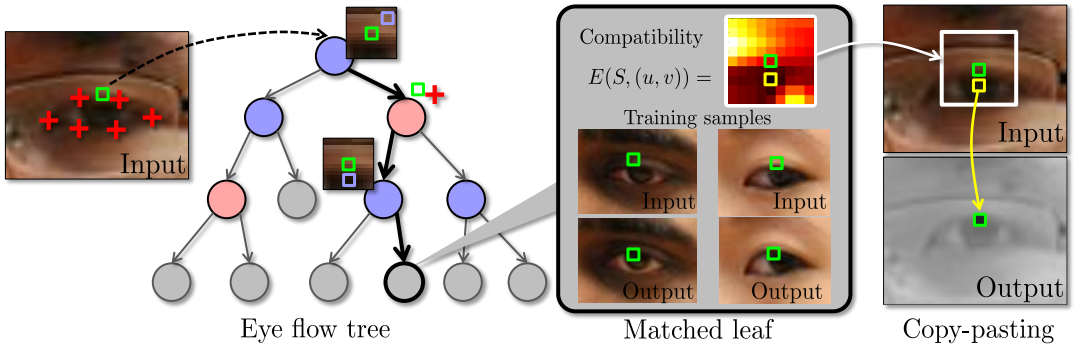


FIGURE 3.2: **Processing of a pixel (green square) at test time in an warping flow tree.** The pixel is passed through an warping flow tree by applying a sequence of tests that compare the position of the pixels w.r.t. the feature points (red crosses) or compare the differences in intensity with adjacent 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.

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 $v = v(x/\Delta, y/\Delta)$, 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 $i \in \{1, \dots N\}$, $\mathbf{l}_i = (l_i^x, l_i^y)$ 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 [Dollár and Zitnick, 2013]). 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 up the unnormalized compatibility score distributions from T trees,one can find a minimizing flow vector with sub-pixel accuracy [Bailey, 2003]. The distribution in the neighborhood of the minimizing pixel is approximated with a parabola using values in the pixel and its 4 neighbors. The parabola minimum is thus the distribution minimum estimated with sub-pixel accuracy. If i is the x-coordinate of the minimum of the discrete distribution $H(x)$ and $H(i-1) \neq H(i+1)$ (otherwise the minimum is exactly at $x = i$), then the sub-pixel minimum coordinate is

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

and the same is true for the y-coordinate.

The warping flow forest system is trained for a specific angular difference. If needed, multiple separate models for different discretized angular differences can 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 produces convincing results for a typical distance between the person and the laptop and typical laptop sizes, so I focus on this angular difference in the experiments.

The images in the Columbia dataset [Smith et al., 2013] 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 [Xiong and De la Torre, 2013] 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).

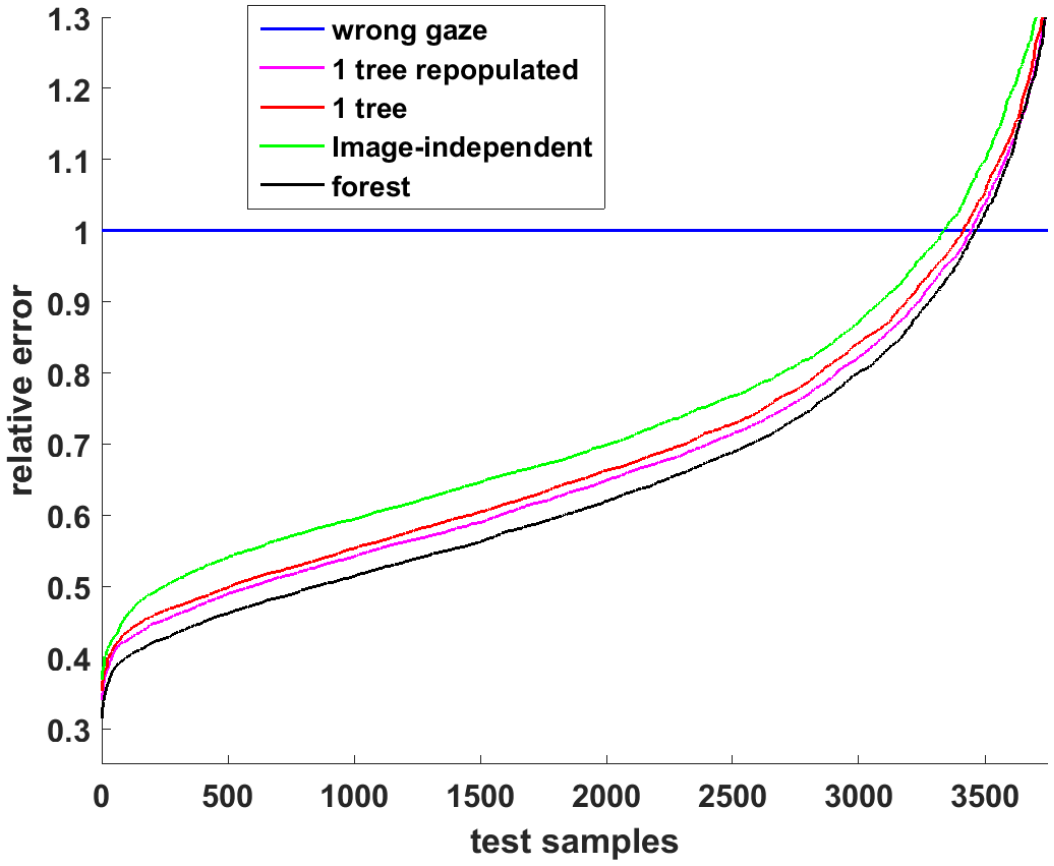


FIGURE 3.3: Quantitative evaluation on the testing sets in the eight-fold experiment: ordered normed MSE errors between the result and the "ground-truth". I compare the one tree version of suggested method (red), the six trees (black), one tree with repopulation (magenta), and the image-independent flow 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.

Quantitative evaluation. I provide a quantitative assess of suggested method on the Columbia gaze dataset [Smith et al., 2013]. 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 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

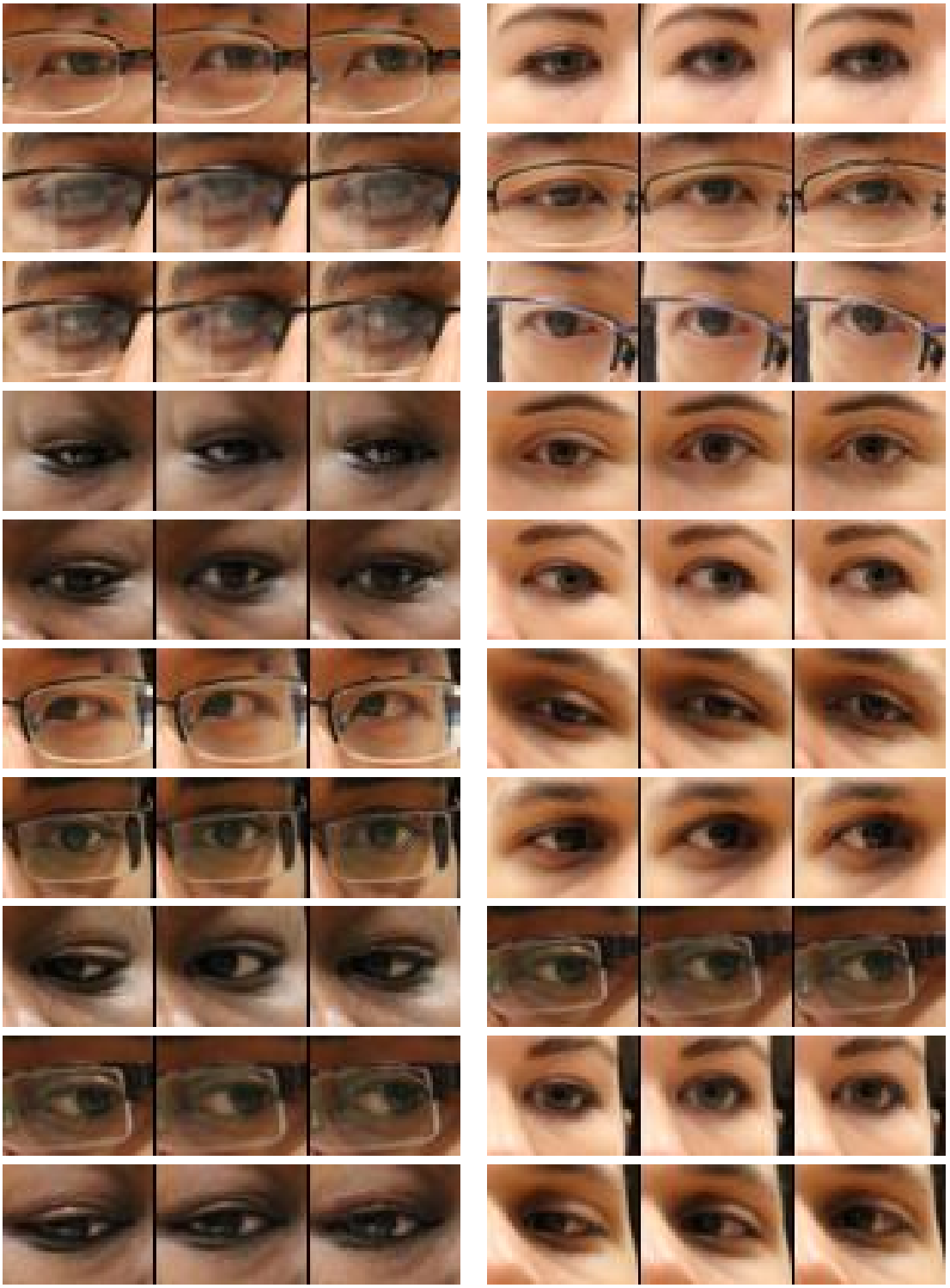


FIGURE 3.4: Randomly sampled results on the Columbia Gaze dataset. In each triplet, the left is the input, the middle example is the "ground truth" (same person looking 10 degrees higher). The right image is the output of warping flow forest. A stable performance of the method across demographics variations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

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.

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 [Kononenko, 2015] 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). 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 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.

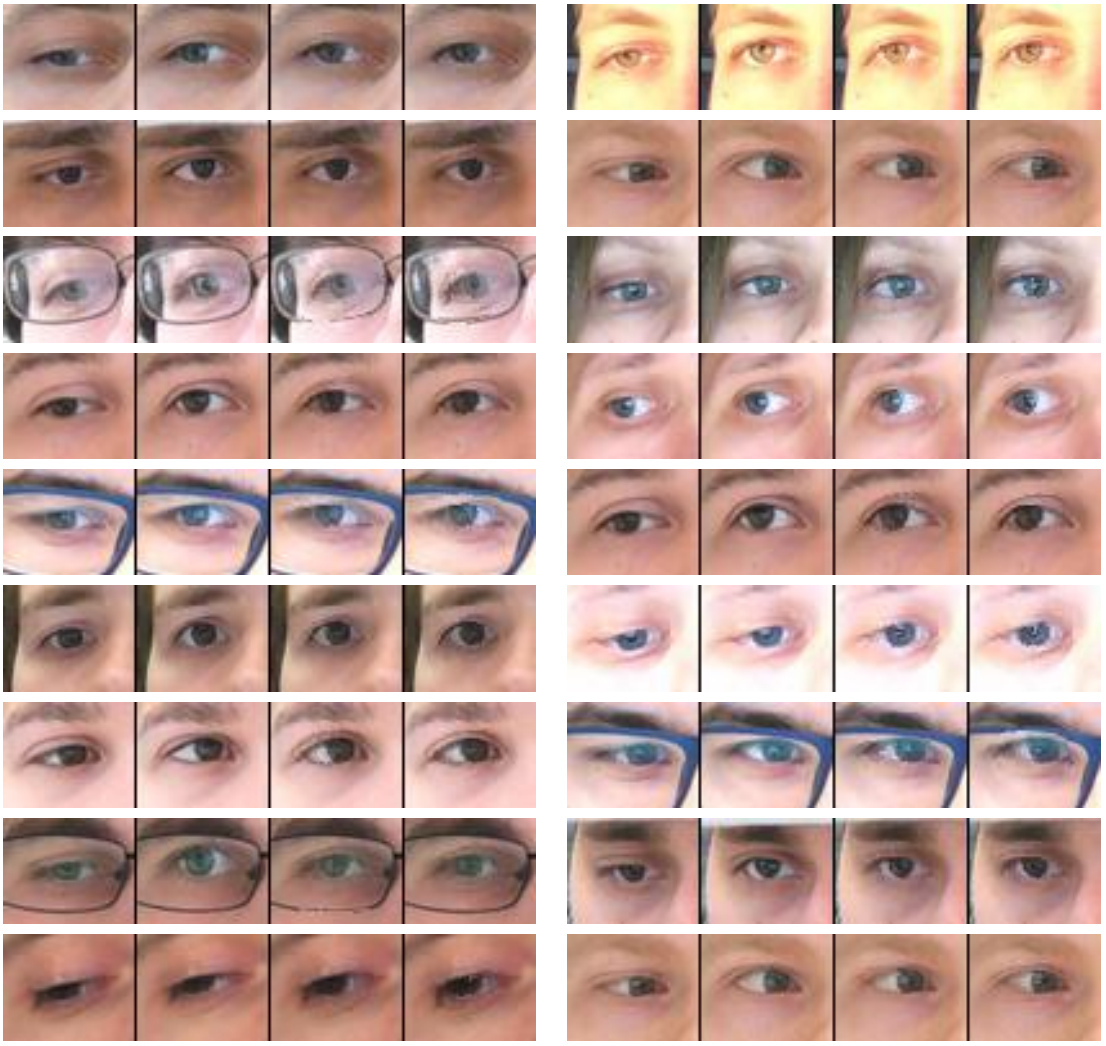


FIGURE 3.5: Randomly-sampled results on the Skoltech dataset (redirection by 15 degrees). In each tuple: (1) the input image, (2) the "ground truth", (3) the output of warping flow forest, (4) the output of image-independent field. Zoom-in recommended in order to assess the difference between the two 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).

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.

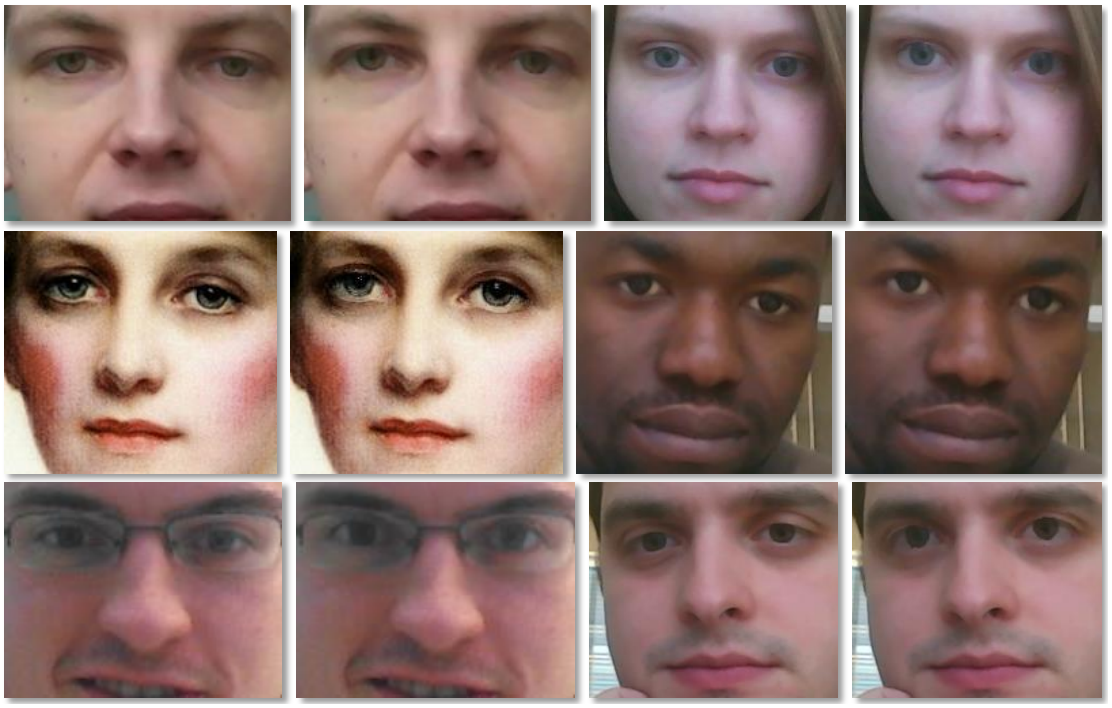


FIGURE 3.7: Qualitative examples of suggested system (based on six trees) showing the cutout of the frames of a video stream coming from a webcam (left – input, right – output). In the first two rows gaze is redirected by 10 degrees upwards, in the third – by 15 degrees. The system induces subtle changes 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.

Chapter 4

Image re-synthesis using deep warping architecture

In this chapter, I proceed to the discussion of the second system (*deep warp*). 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 t predict the warping field. This field is predicted in two stages in a coarse-to-fine manner, where the decisions at the fine scale are informed by the coarse stage result. Beyond coarse-to-fine warping, the photorealism of the result is improved by performing pixel-wise correction of the brightnes, where the amount of correction is again predicted by the network (Figure 4.4). All the 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 the eye appearance before and after 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 share of the work presented in this chapter was done by my collaborator Yaroslav Ganin. My personal contributions include the experimental setup and data preparation for the Deep Warp approaches, comparison of the Deep Warp and forest-based approaches, and setup and 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 at new points, i.e.

find f^* – an approximation to f which is close to f at 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:

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

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

Typically, the layers are rather simple nearly linear functions. However, the composition of linear functions is also a linear function. A common design motivated by 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 one-dimensional function. Typical choices of activation 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, whose 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 the 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. The choice of a sigmoid function implies thinking of the output as a probability of a Bernoulli trial. A 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, a large data set, a non-convex cost function and possible overfitting. A common approach to optimization is 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 back-propagation [Rumelhart et al., 1988], a method that involves clever bookkeeping of partial derivatives by dynamic programming. To deal with the large dataset and avoid redundant computations, it is sensible to apply another standard approach, stochastic gradient descent (in particular, mini-batch gradient descent) which performs 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 [Nesterov, 1983] and Adaptive Moment Estimation [Kingma and Ba, 2014].

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 is convolutional networks which can be applied to data with a special grid-like topology, for example images, making them extremely useful in computer vision [LeCun, 1989, Krizhevsky et al., 2012]. Convolutional networks are neural networks where convolution is used instead of general matrix multiplication in at least one layer. As compared to the fully connected layer (4.2), the convolutional layer has much fewer parameters, because the convolution parameters depend on the layer only and not on a particular location in the feature map. This does not only reduce overfitting but also makes the network 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 the neighborhood, thus providing independence of small shifts in the input.

The batch normalization layer [Ioffe and Szegedy, 2015], used in this work, is a means of reparametrization that helps to mitigate the issue of coordinating updates across many layers. At training time, each mini-batch of layer activations is normalized by subtracting the mean and dividing it by the standard deviation. Importantly, back-propagation is performed 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, the running average mean and deviation from the training time are used.

[Jaderberg et al., 2015] introduce an attention mechanism based on Spatial Transformer Networks. The authors propose a method with a specific part of the network producing the grid that the input image or feature maps will be sampled from. In the case of the pixel-wise replacement approach suggested in this work (Section 2.1), the warping field (2.4) is a grid with the same functions. [Jaderberg et al., 2015] show how this operation can 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, and 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 [Ioffe and Szegedy,

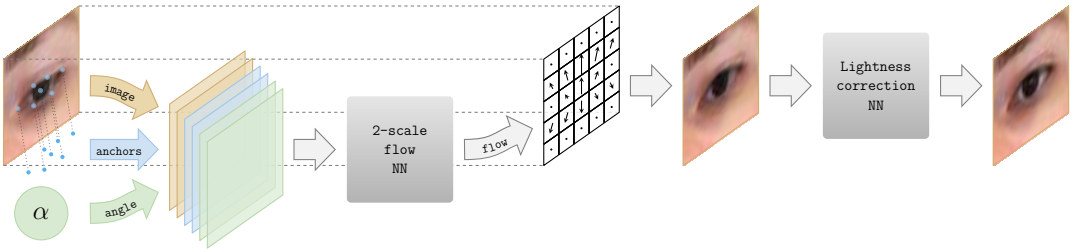
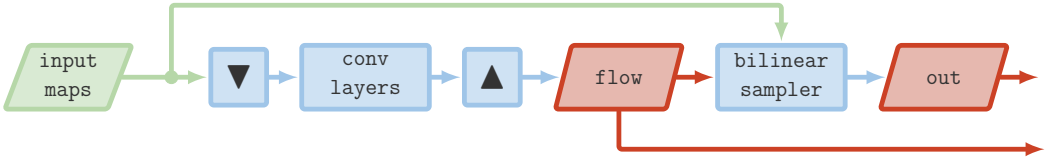
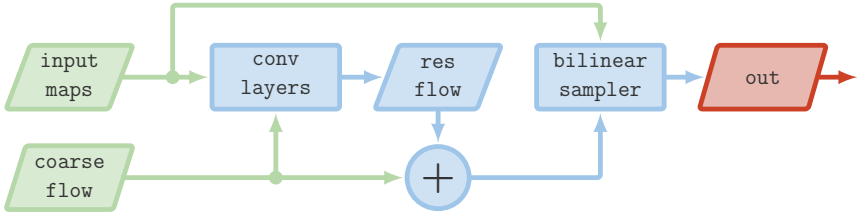


FIGURE 4.1: The deep warp system takes an input eye region, feature points (`anchors`) as well as a correction `angle` α and sends them to the multiscale neural network (see Section 4.2) predicting a `flow` field. The flow field is then applied to the input image to produce an image of a redirected eye.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

2015] and ReLU nonlinearities (the actual configuration is shown in Figure 4.2(c)). To preserve the resolution of 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 $\mathbf{S}$

[Jaderberg et al., 2015] 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 [He et al., 2016]):

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

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 [Ghodrati et al., 2016], the correction angle is treated as an attribute and is embedded into a higher dimensional space using a multi-layer perceptron $\mathbf{F}_{\text{angle}}(\alpha)$ with ReLU nonlinearities. The precise architecture is `FC(16)` $\rightarrow$ `ReLU` $\rightarrow$ `FC(16)` $\rightarrow$ `ReLU`. Unlike [Ghodrati et al., 2016], 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.

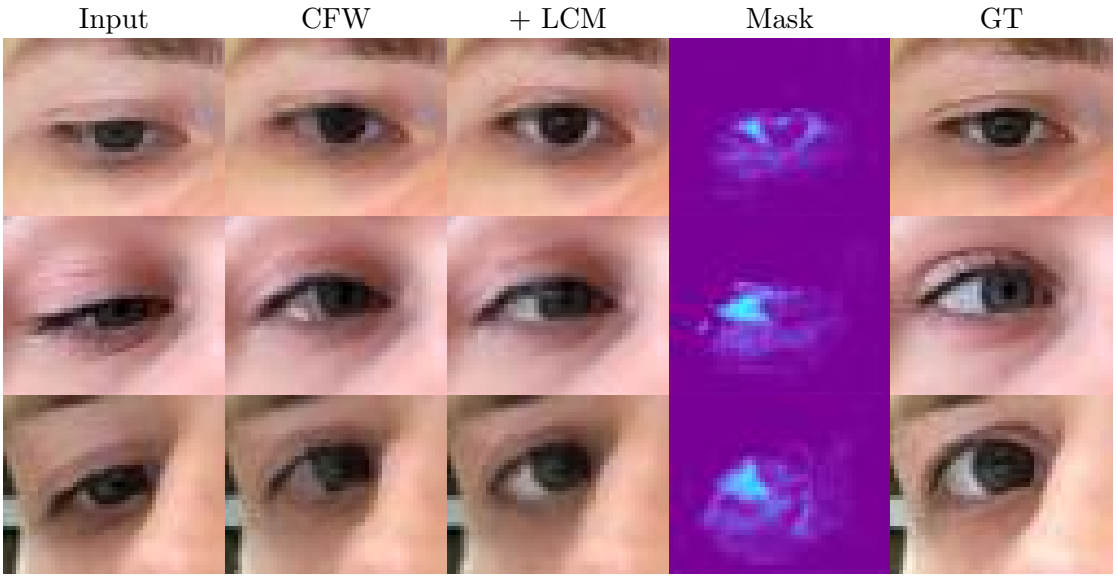


FIGURE 4.3: Visualization of three challenging redirection cases where the **Lightness Correction Module** helps considerably compared to the system based solely on coarse-to-fine warping (CFW), which is having difficulties with expanding the area to the left of the iris. The 'Mask' column shows the 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:

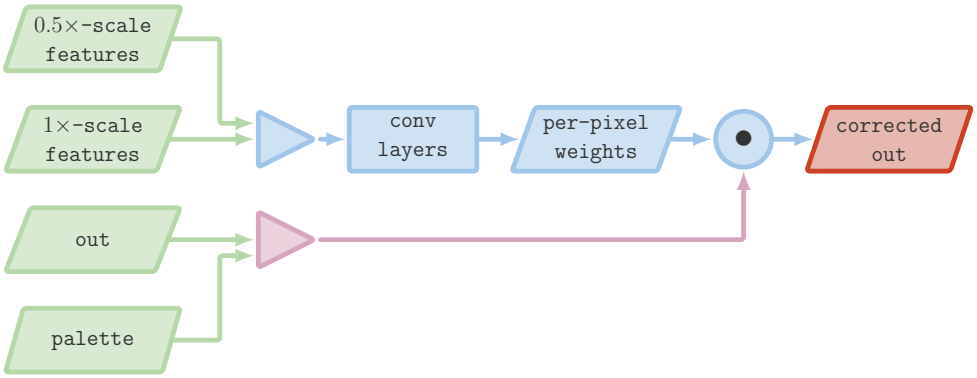
$$\begin{aligned} \Delta_x^i[x,y] &= x - l_i^x, \\ \Delta_y^i[x,y] &= y - l_i^y, \end{aligned} \quad \forall (x,y) \in \{0,\ldots,W\} \times \{0,\ldots,H\},$$

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.

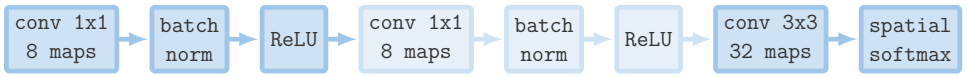
Ultimately, the input map stack consists of 33 maps (RGB + 16 angle embedding maps + 14 feature point embedding maps).

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. This constraint on the refinement operation represents the desire to compensate the lack of white in the results of warping (see Figure 4.3). 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 [Kingma and Ba, 2014]. 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.

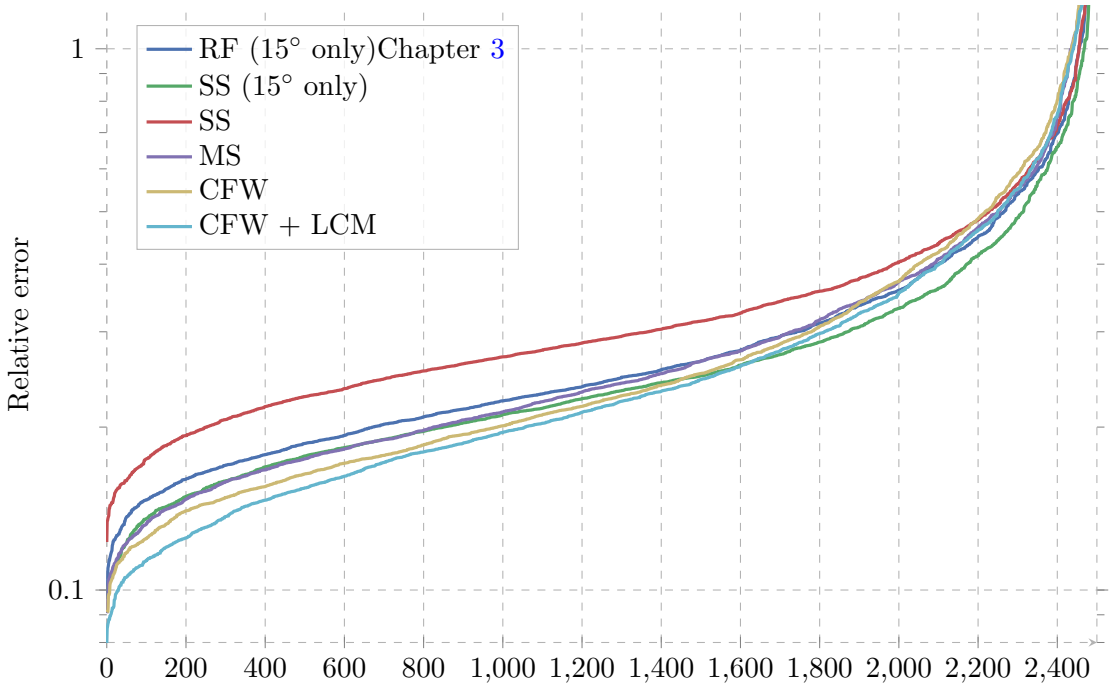


FIGURE 4.5: Ordered errors for 15° redirection. Multiscale models (MS, CFW, CFW + LCM) show results that are comparable or superior the Random Forests (RF) Chapter 3.

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

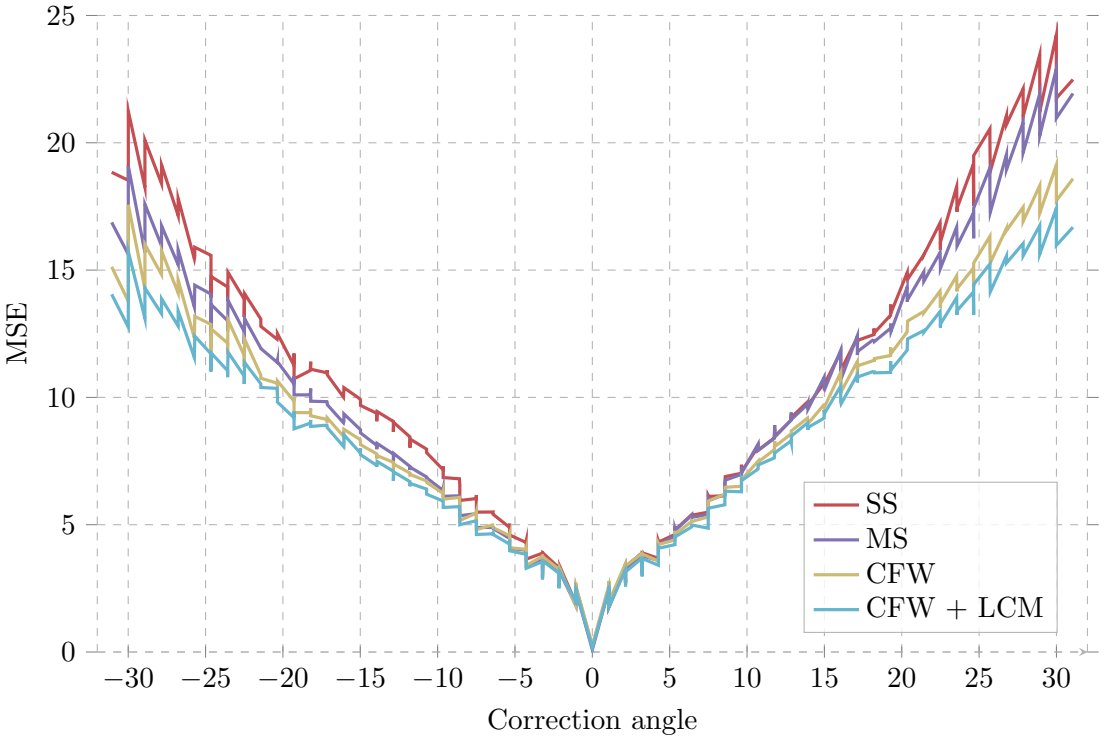


FIGURE 4.6: Distribution of errors over different correction angles.

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 of 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

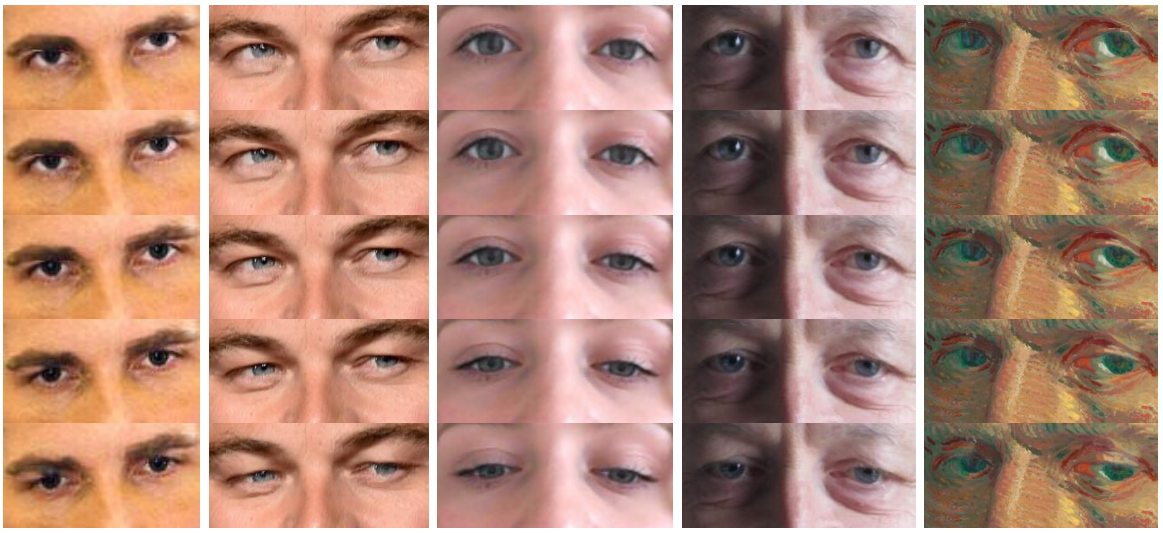


FIGURE 4.8: Gaze redirection with deep warp model trained for vertical gaze redirection. The model takes an input image (middle row) and the desired redirection angle (here varying between -15 and +15 degrees) and re-synthesize the new image with the new gaze direction. Note the preservation of fine 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 deep warping system suggested in Chapter 4 compares well with the warping flow forest-based system presented in Chapter 3 in terms of quality of the results. Another drawback of forest-based system is its big memory footprint proportional to the patch size, which is required to store distributions of the compatibility score (3.12) in the leaves. In the implementation with 10 trees and 9-by-9 patches, the resulting model has a size of 200Mb.

Better quality of the deep warp results is attained at the expense of much higher computation time (few frames per second on a GPU). In this chapter, a system based on *neural network-supervised forests* is proposed and aiming to combine 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 deep warp predictions 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 forest to emulate the predictions of the neural network relates this work to several recent papers with similar teacher-student architectures. The idea to use the output of a very precise but large and slow model as supervision for a faster architecture goes back as far as [Bucilu et al., 2006]. They show that the model ensemble can be compressed in a single much faster model.A different kind of training is often

needed to transfer knowledge from the teacher model to a smaller model. The idea is to pass unlabeled data through a large, accurate model to collect the scores produced by that model. A fast and compact model does not overfit and approximates well the target function, provided enough pseudo data by high performing model ensemble. [Bucilu et al., 2006] suggest several ways of generating new pseudo data, estimating the joint distribution of attributes in the original training set for the large model and drawing points from this generative model trained on the original training set.

An alternative way is to train a smaller model on the original training points only, while training it to copy other features of the model, such as its posterior distribution over the full set of classes, is suggested in [Hinton et al., 2015]. The authors suggest training a "student"network on the softened output of an ensemble of wider networks (teacher networks), thus allowing the student network to capture both the information provided by the true labels and the finer structure learned by the teacher network. This way an ensemble of deep networks is compressed into a student network of a similar depth. In particular, [Hinton et al., 2015] use a softened version of softmax teacher output

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

with temperature $T > 1$ (the case $T = 1$ corresponds to the usual softmax). The student network is trained 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 in much the same way as in 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 [Romero et al., 2015] 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 a low enough depth and a 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. An additional layer is introduced in the student network to regress the middle layer of the teacher network from the middle layer of the student network. The corresponding error gradient in this architecture affects the student networks lower layers only. Thus the objective is two-fold: help the higher layers of the student network to predict the label and model the intermediate layer of the teacher network. It is shown in the experiments that such hints on the middle layers help the student network (which learns very poorly without hints) to improve its results both on the train and test sets.

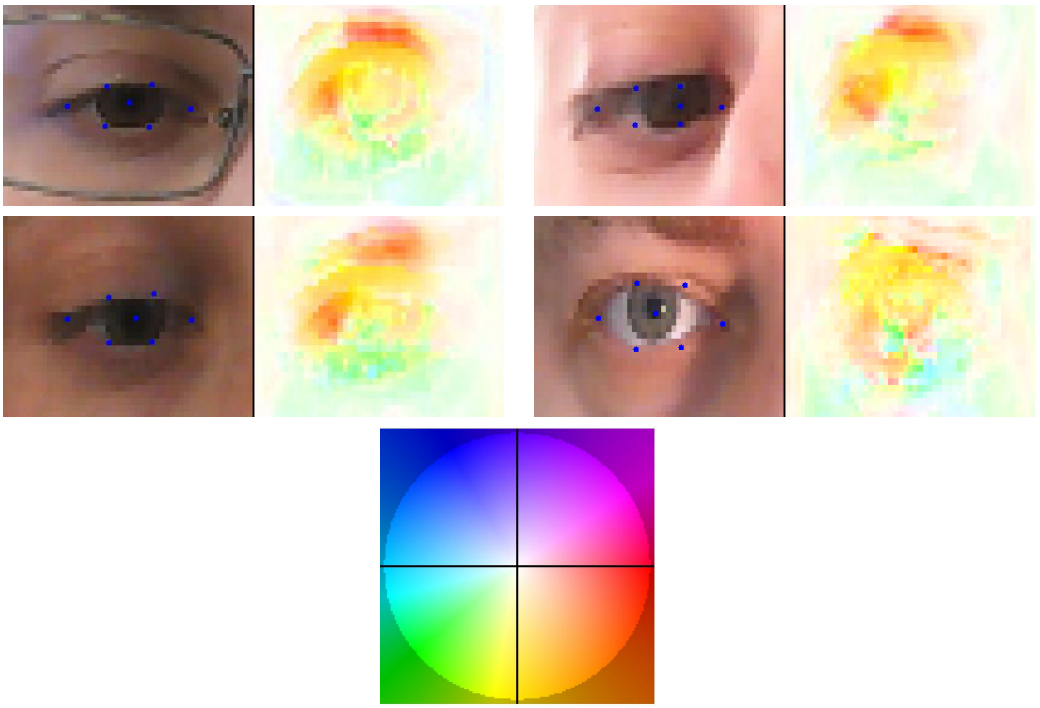


FIGURE 5.1: The output flow of the warping modules (Section 4.2) on random samples from a training set. The coarse-to-fine model without lightness correction was trained on a task of 15° redirection upwards. This data is used to train a neural network-supervised regression random forest. The bottom figure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

5.2 Learning

To learn the system, I fix the desired redirection angle and use the sum of the coarse and fine flows 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 falling within 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 the flow is limited, only two bytes are sufficient to store a flow vector in each leaf, which is much less as compared to a weakly-supervised forest. Hence the reduction in memory is $(w^2 - 2)$ bytes for each leaf, where w is 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 decreased, resulting in typical memory demand for trained forests of only three megabytes.

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 the 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 the output in this case is a one-dimensional vector.

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

5.3 Experiments

In this section the neural network-supervised forests are compared with the weakly-supervised forest from Chapter 3 and the deep warping approach from Chapter 4 in

terms of performance.

5.3.1 Evaluation using Mean Squared Error

The methods are evaluated on the Skoltech dataset. The initial set of subjects is randomly splitted into the training and testing sets and image pairs are sampled applying the same pre-processing steps as during data preparation (Section 2.3).

15° **correction.** First a comparison of MSE errors for the 15° vertical gaze redirection case is offered. The models considered for this comparison are:

1. A system based on Weakly-Supervised Random Forests (*WSRF*), described in Chapter 3. The score of this model is what random forest can achieve without supervision from the neural network.

2. Deep Warp system Chapter 4 without lightness correction (*Deep Warp withou LCM*) 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 (*NNSRF*) described 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 the train set).

4. A Deep Warp coarse-to-fine warping-based system with a lightness correction module trained for 15° vertical gaze redirection (*Deep Warp*) (Section 4.4), used to test the effect of the lightness correction model on the score.

5. Simple baseline: Image-Independent Flow field (Section 3.2.1) where the flow is based solely on the relative position of the pixel. The score of this model is in some sense the basic score of the warping approach, i.e. what we can get without any grouping of similar pixels or representation learning.

Figure 5.2 contains a graph of sorted normalized errors where all the 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 the NN-supervised forest performs nearly as well as 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 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. These are mostly the cases similar to those shown

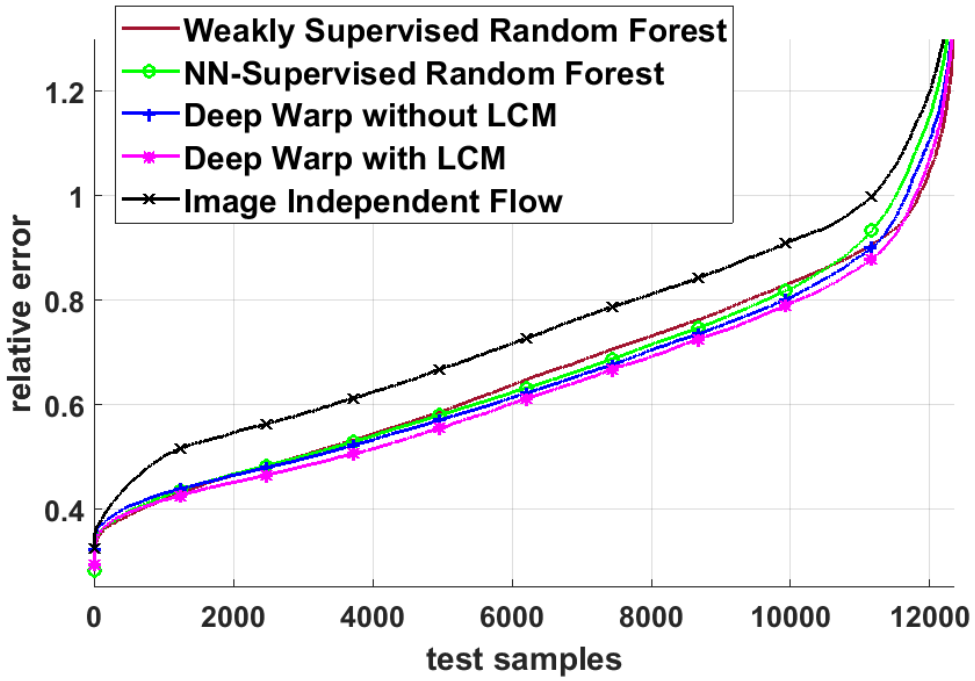


FIGURE 5.2: Ordered errors for 15° vertical gaze redirection (see the text for further discussion of the error metric). The best performance is shown by the full Deep Warp architecture with the lightness correction module. Marks on the curves do not denote data points, but are plotted for distinguishing the curves in case of black and white printing.

in Figure 4.3. Unified multiscale models trained to handle different angles (not included in this comparison) are generally comparable to the one tailored for 15° redirection. It is also worth noting that even a single-scale model trained for this specific correction angle consistently outperforms the 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 about whether the gaze difference between the input image and the output equals the requested angle. One could imagine an example (possibly degenerate) where 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 required. Therefore, I provide an additional assessment to evaluate the redirection angle on the test set.

Firstly, a redirection angle evaluation tool should be developed. I use the evaluation model E trained to determine the angular difference between two input images of the eyes. For the model to be more accurate, I consider vertical redirections only. The model is parametrized as a deep neural network. The inputs to the 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 the 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 in random manner too (the maximum vertical gaze difference in the database is 36°, Section 2.3). The mean RMSE is 0.8° on the training set and 1.1° on the validation set.

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

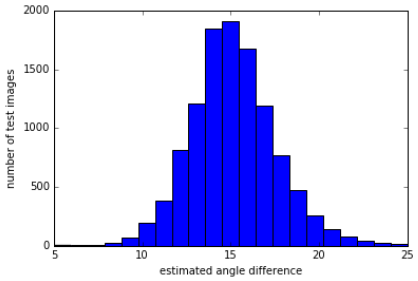
$$\mathrm{conv}(48, 3) \to \mathrm{conv}(48, 3) \to \mathrm{MaxPool} \to \mathrm{conv}(48, 3) \to \mathrm{conv}(48, 3) \to \mathrm{MaxPool}.$$

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

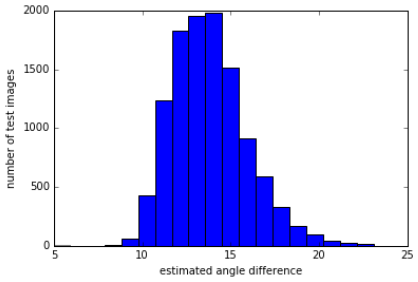
$$\mathrm{conv}(96, 3) \to \mathrm{conv}(96, 3) \to \mathrm{MaxPool} \to \mathrm{conv}(96, 3) \to \mathrm{conv}(96, 3) \to \mathrm{MaxPool}$$
$$\to \mathrm{FC}(1000) \to \mathrm{FC}(250) \to \mathrm{FC}(1),$$

where the last activation is linear.

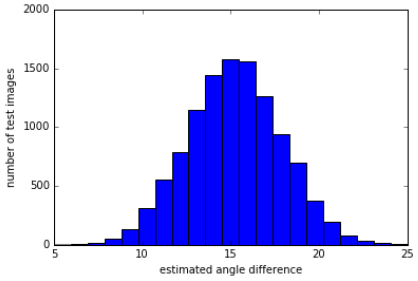
This model was used when evaluating the results of 15° upward redirection on the testing set. For each input image I, the resulting image O_m with gaze redirected by 15° upwards was obtained by each method. Then, the actual redirection angle was estimated as $\alpha \approx E(I, O_m)$. The results of this assessment are presented in Figure 5.3. The distribution of the outputs produced by the 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 of the dataset labels and imperfection of the evaluation network. The mean of distributions for all the models is quite close to 15°, while the variance does not exceed that of the distribution corresponding to the ground-truth pairs. Thus I conclude that all the methods performed well in this test, which suggests that they should be compared based primarily on the realism of their results. Interestingly, the best performance was shown by the Deep Warp model with no lightness correction module, although LCM decreases the ℓ_2 error. The means of the forest-based models are slightly less than 15°, while the mean of CFW+LCM model is slightly higher than 15°.



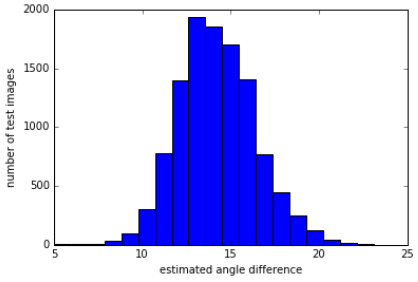
Distribution corresponding to the pairs with 15° difference according to the data collection procedure. Mean = 15.1°, std = 2.6°.



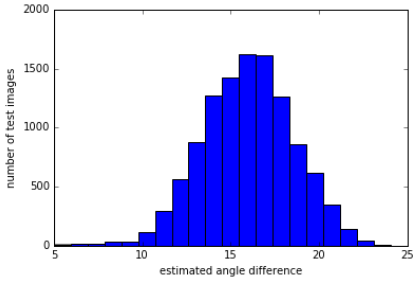
Weakly-supervised random forest (Chapter 3). Mean = 13.8°, std = 2.1°.



Deep Warp without LCM (Chapter 4). Mean = 15.2°, std = 2.7°.



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



Deep Warp 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 reference, the distribution for 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 with no lightness correction model.

5.3.3 Qualitative evaluation

To make a qualitative comparison of the systems, I show a *random* subset of results of redirection by 15 degrees upwards in Figure 5.4. All the methods were trained for 15° vertical redirection. One can see that the system is able to learn to redirect the gaze of unseen people rather reliably, attaining a close match with the ground truth. Deep warp systems produce results with better visual similarity to the ground truth as compared to forest-based systems. The effect of lightness correction is quite obvious: the system with a lightness correction feature performs clearly better on the input image with the sclera invisible in one corner. On the downside, lightness correction can result in 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. Figure 5.7 illustrates the results of the NNSF-system for lower input resolutions on the images randomly sampled from a test set. For that, images are downsampled from the original sizes of 80×100 to 10×12, 20×25, and 40×50 respectively. The NNSF-system is then applied to the downsampled versions. The effect of gaze redirection is noticeable even at very low resolutions.

Comparison with the previous work on Gaze Correction. Figure 5.8 I gives a side-by-side comparison with the system [Giger et al., 2014] which also performs monocular gaze correction for the videoconferencing scenario. The difference in the approaches is clearly visible. The method proposed here redirects the gaze and confines the changes to the eye region, while keeping the head pose unchanged. By contrast, the system [Giger et al., 2014] synthesizes a novel view for the facial part and then blends the new face area into the input image. The latter approach results in certain distortion of face proportions. Moreover, [Giger et al., 2014] requires a simple per-person pre-calibration step and 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).

FIGURE 5.4: Results on a random subset of the hold-out test set. From left to right: (a) Input, (b) Weakly Supervised Random Forest, (c) Neural Network Supervised Random Forests, (d) Deep Warp with the lightness correction module, (e) Deep Warp without the lightness correction module, (f) Ground truth. The full variant of Deep Warp system (d) generally performs the best.

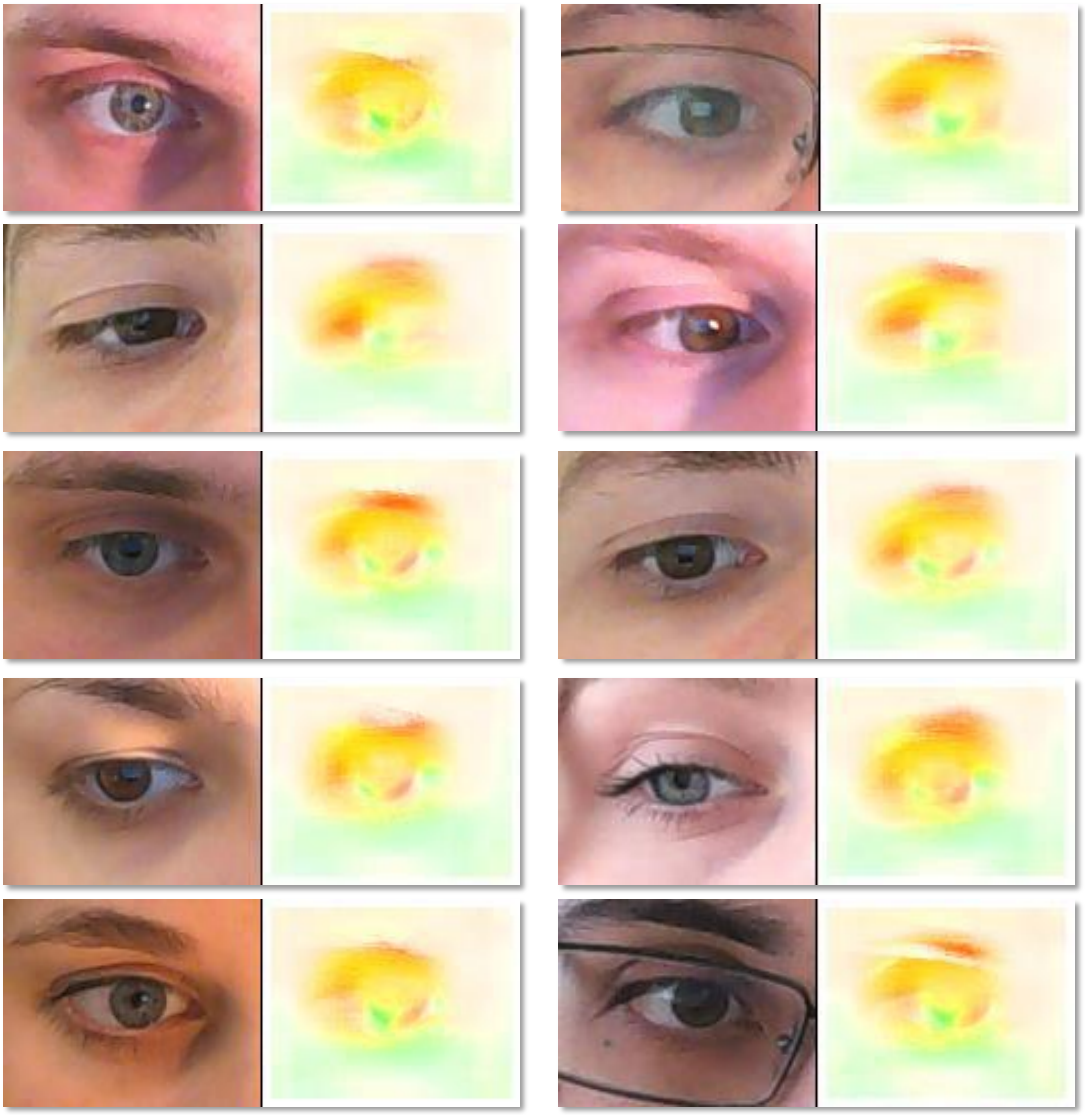


FIGURE 5.5: Examples of warping field taken randomly from a validation set. In each pair, the left image is the input and the right one is the predicted warping field. The color pattern is the same as in Figure 5.1.

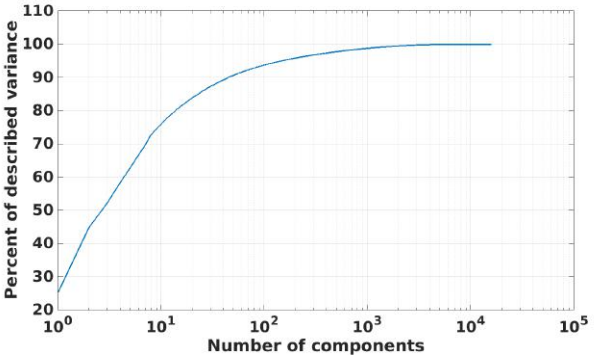


FIGURE 5.6: Inherent dimensionality of the warping fields manifold. The dependence of the percentage of explained variance on the number of PCA components is plotted. The scale of the x-axis is logarithmic. One hundred components describe almost 95% of the variance.

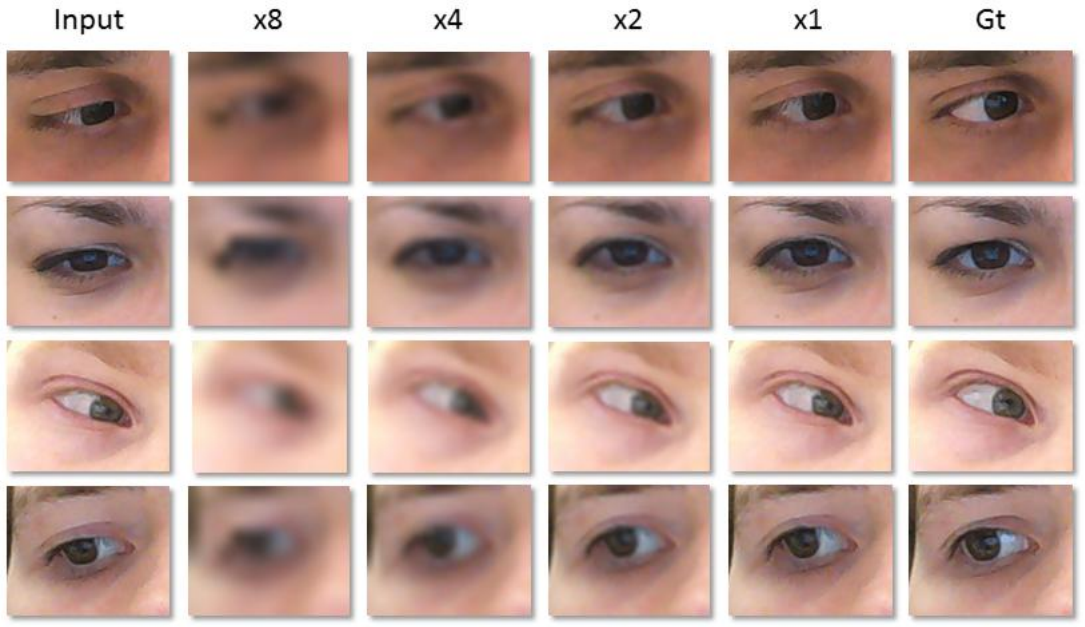


FIGURE 5.7: The results of NNSRF system for the 15° upwards redirection with input images of lower resolutions (the downsampling factors are shown at the top). Gaze redirection is persistent even for very low resolution images.

Finally, the supplementary video [Kononenko, 2017] demonstrates a real-time screencast of NNSRF system running on a variety of people under a variety of conditions typical for the teleconferencing scenario.

5.3.4 User study

To confirm the perceptual improvement in the results, a user study similar to the one presented in Section 4.6.2.1 is performed. Here the setup is changed to show the full face image, while still telling the users that only the eye region will be re-synthesized. Showing the full face was driven by the need to capture the users perception of the overall realism of the resulting images in a more straightforward way.

The user study was performed on 41 subjects unrelated to computer vision in order the compare the four methods (WSRF, NNSRF, Deep Warp without LCM, Deep Warp with LCM). Each user was shown 80 quadruplets of images, each containing three unprocessed real face images and one image obtained by re-synthesis using one of the methods. A screenshot of the user study interface is shown in Figure 5.9.

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 shown a quadruplet, the subject was supposed to click on the artificial (re-synthesized) image as quickly as possible. For

FIGURE 5.8: Comparison with monocular gaze correction method from [Giger et al., 2014]. **Left** – the input video frame (taken from [Giger et al., 2014]). **Top right** – output of the suggested NNSRF system. **Bottom right** – result of [Giger et al., 2014]. While both systems achieve a convincing redirection effect, NNSRF system avoids distortion of facial proportions (especially in the forehead and chin regions), while also not requiring a GPU to achieve real-time performance.

each method, the number of correct guesses out of 20 is then recorded (the expectation would be 5 for an ideal method and 20 for a very poor one). The time it took the subject to make a decision on each quadruplet is also recorded (a better method would take a longer time for spotting). The results of this experiment are summarized in 5.1.

In general, all the methods performed very well, closely approaching the highest 25% mark (random guess performance). The NN-supervised random forest and full deep warp system displayed comparable performance, while the other two (warping flow forest and coarse-to-fine warping without lightness correction) did slightly worse. However, considering the fast (confident) clicks only, neural networks are somewhat inferior to the warping flow forest and NN-supervised forest. The deep architectures behavior can be put down to fixed basic resolution, as opposed to the forest-based methods capable of processing eye images at the cropped images native resolution. In some instances, the

	WSRF	**NNSRF**	**Deep Warp without LCM**	**Deep Warp with 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.

users could apparently quickly notice the interpolation artifacts or the artifacts near the border in the deep warp system results. As a consequence, the teacher CFW method performed a bit worse than the student NNSF in these metrics. If assessed in terms of the mean time it took a user to make a guess, the deep warp architectures outperformed the forest-based systems thanks to the hardest samples where users got stuck for a long time. However, it might be well to point out that the user results display a very high variance, simply because some people are much more attentive than others. Increasing the number of people would not fix the problem, for there is still a significant amount of very good and very bad results. What could help though is running a very large number of tests for each user but users are unlikely to agree to being tested for hours without proper motivation. Thus, for example, it is not so easy to analyze the test results by applying statistical tests.

I also present some comparisons for pairs of methods. In the plots in Figure 5.10, the

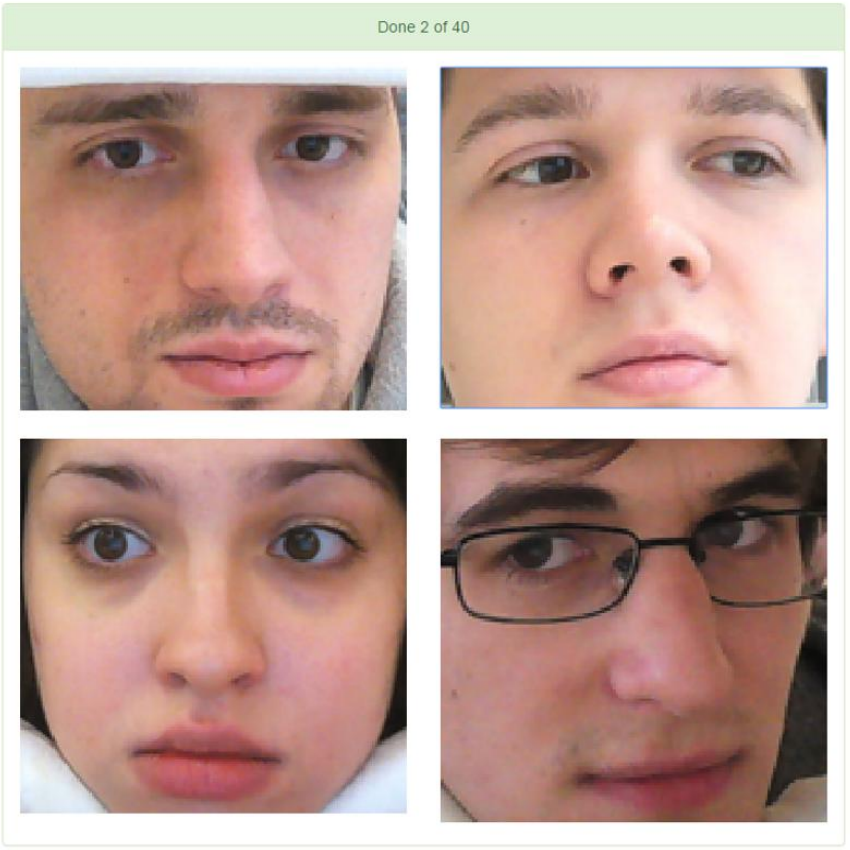


FIGURE 5.9: Screenshot from the user study interface. The user was told that one of the four images was not real and was asked to click on the one that looked the most unnatural, spending as little time as possible. The right answer in this example is the top left image.

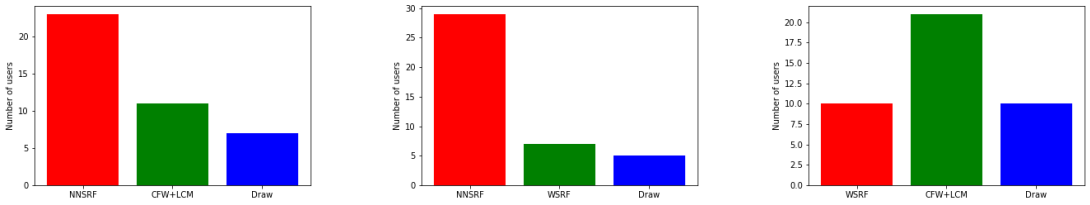


FIGURE 5.10: User study results for pairs of methods. Red bar number of users fooled more often by the first method, green bar by the second method, blue bar shows number of draws. For example, in the middle plot, the neural network supervised forest outperformed the weakly-supervised forest.

red bar shows how many users were fooled more often by the first method, the green bar by the second method and the blue bar shows number of draws. The NN-supervised random forest looks best 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 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 corresponds to the case 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 by half will result in nearly two-fold speedup and only negligible quality degradation).

The neural network supervised forest method performs nearly as well as 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 the warping flow forest due to a smaller tree depth, however this difference is not crucial in my implementations. A significant improvement is observed in memory consumption: the warping flow forest method typically requires $100-200$Mb RAM storage at test time, while the neural network supervised forest requires only $1.5-3$ Mb. There is indeed a whale of difference in memory requirements, because error distributions are stored in the leaves in the first case (Section 3.2) and only the two-dimensional flow vector is stored in the latter case (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 and 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 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 taking multiple images of people in a constrained setting (in a literal sense – c.f. Figure 2.5 and 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. Collecting such data is much easier than in the fully-supervised case. To be more precise, the model presented here uses unsupervised manifold learning to construct a deep embedding of eye images into a low-dimensional latent space (*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 complete, the system can redirect the gaze of an arbitrary eye image by traversing the manifold in a latent space. In particular, the image is mapped to latent space and then the 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 from [Reed et al., 2015], although 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 a known difference in gaze direction) to estimate the displacements in latent space that are typical for certain gaze redirections (e.g. lifting the gaze by 15 degrees which is relevant for the videoconferencing scenario).

As experiments in Section 6.4 show, the resulting semi-supervised solution achieves convincing gaze redirection,displaying better visual quality than the fully supervised solution of the fully supervised deep warping method (Chapter 4) in the case of training data shortage. As compared to the model of [Reed et al., 2015], using warping rather than direct re-synthesis ensures high realism of the resulting images and prevents the loss of high-frequency details. Section 6.1 gives an overview of related approaches, Section 6.2 addresses the bulk of the model and explains how it can be trained on unsupervised data and Section 6.3 shows how the model can be trained using a small amount of eye images with known absolute or relative gaze direction.

6.1 Related work

The idea of image analogies goes back as far as [Hertzmann et al., 2001]. 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 and C. The approach put forward in [Hertzmann et al., 2001] is first to compute Gaussian Pyramids for given images and then construct a pyramid for D such that the neighborhoods are similar to the one generated from B, while having similar multi-resolution appearances at the corresponding locations in A and C. This approach is applied to image filtering, texture synthesis and transferring, super-resolution.

Related to a more general deep-analogy making model from [Reed et al., 2015]], the proposed model leverages on the ability of making analogies by addition and subtraction (Figure 6.1) which was applied to word embeddings in [Mikolov et al., 2013, Pennington et al., 2014]. The model in [Reed et al., 2015] is trained on tuples (a, b, c, d) such that $a \; ? \; b :: c \; ? \; d$ and at test time is capable of producing an unknown d – the result of applying the transformation $a : b$ to c. Encoder $f : \mathbb{R}^D \rightarrow \mathbb{R}^K$ and decoder $g : \mathbb{R}^K \rightarrow \mathbb{R}^D$ are parametrized as deep convolutional neural networks trained end-to-end 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 of this transformation vector to $f(c)$.

The suggested objective for addition-based analogy making is

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

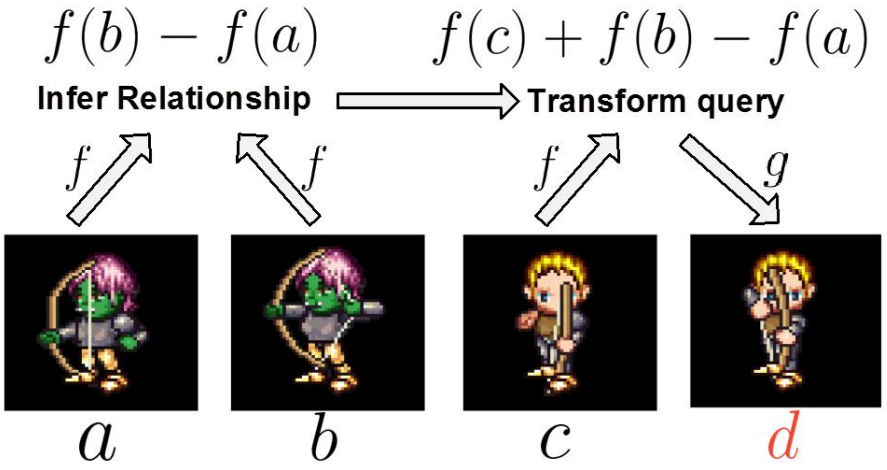


FIGURE 6.1: **Visual analogy making concept.** An encoder function f maps both a query image and an analogy pair onto an analogy-making space, where the space analogy is some simple operation, for example, summation. The decoder retrieves the result back to the image space. Figure taken from [Reed et al., 2015].

Since the vector-addition approach is sometimes unable to model all the required transformations (e.g. rotation: repeatedly adding the same vector will not restitute the original embedding vector), [Reed et al., 2015] also suggests a 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. To facilitate accurate traversing of the image manifolds in the embedding space, [Reed et al., 2015] introduces regularization which compares the difference between the embeddings of the prediction and query $f(d) - f(c)$ to the predicted increment vector in the 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).

[Reed et al., 2015] also propose a method for incorporating features disentangling into the analogy model. The model is learned on three-image tuples (a, b, c): the first and the second are used to extract hidden units in latent representation and the third is a ground-truth image. Disentangling is implemented by introducing a binary vector of switches $s \in \{0, 1\}^K$ which indicates whether it is the first or the second image that the corresponding feature from the embedding should be taken from.

The model [Reed et al., 2015] needs to know the transformations that should be provided in the form of analogy-forming quadruplets, whereas proposed model does not. More closely resembling proposed model at training time is the inverse graphics model from [Kulkarni et al., 2015] that can be trained at a level of supervision similar to that of the system suggested here, i.e. using subsets of images where some factors of variation are invariable while others vary arbitrarily. The authors also make use of an autoencoder-like architecture with image embedding latent space and present an approach for learning interpretable latent space representations for out-of-plane rotations and lighting variations. Their training procedure is designed so as 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, azimuth and light source angles) and intrinsic properties describing identity, shape, expression, etc. The data is organized into mini-batches in such a way that only one of the extrinsic variables changes across the mini-batch, while all the other extrinsic and intrinsic variables, e.g. rotation of the same object, are fixed. 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 the latent variables except the selected z_{train} with the average across the mini-batch.
5. Project the mini-batch back to the image space using the decoder, calculate the reconstruction error and back-propagate it through the decoder.
6. Replace the gradients for all the latent variables except the selected z_{train} with the difference between the values of the variables and their mean value across the mini-batch.
7. Backpropagate through the encoder.

In this procedure, step 3 ensures that the variation in the selected factor is limited to the selected z_{train} and step 6 equalizes all the other variables across the mini-batch. however tend to produce blurry non-photorealistic images that are not suitable for gaze redirection application scenarios (Figure 6.2). The model suggested in this chapter deals with the issue by applying warping instead of direct re-synthesis.

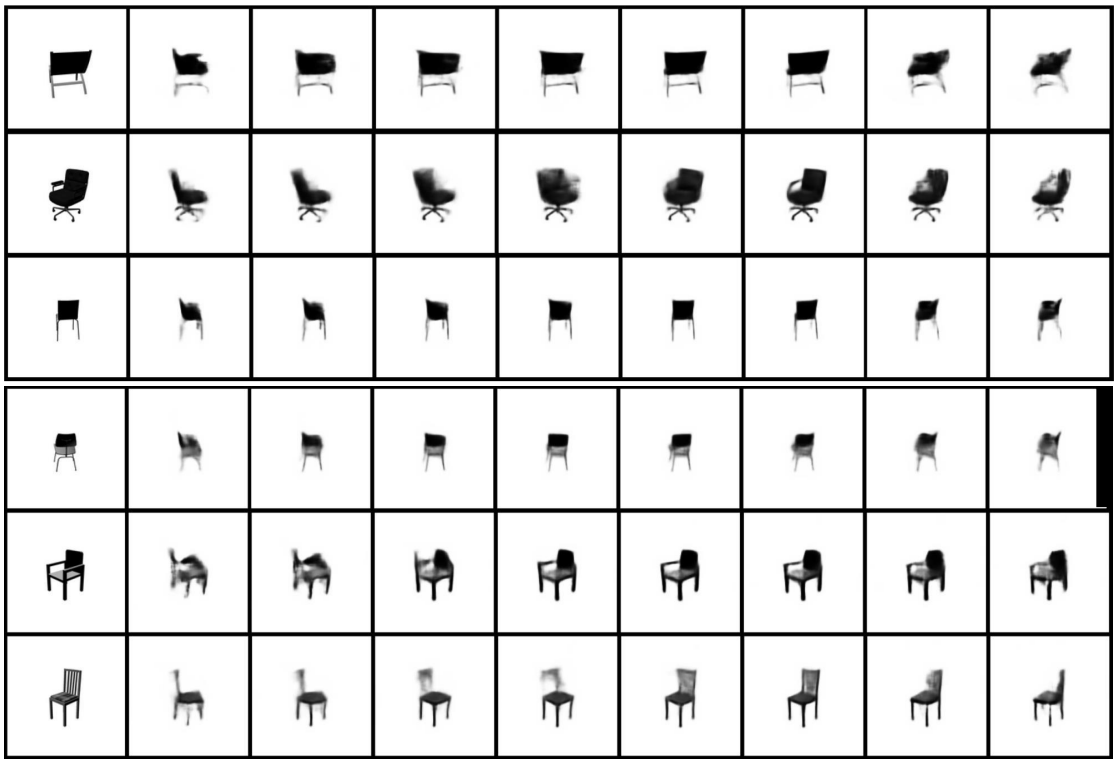


FIGURE 6.2: **Manipulation rotation results from [Kulkarni et al., 2015].** Each row was generated by encoding the input image (leftmost) with the encoder, then traversing the manifold by changing the value of a single latent variable and putting the modified encoding through the decoder. Although the rotation is quite convincing, the suggested approach, as opposed to the warping model, has obvious drawbacks, such as blurring and a loss of fine details (see also Figure 2.2 and 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

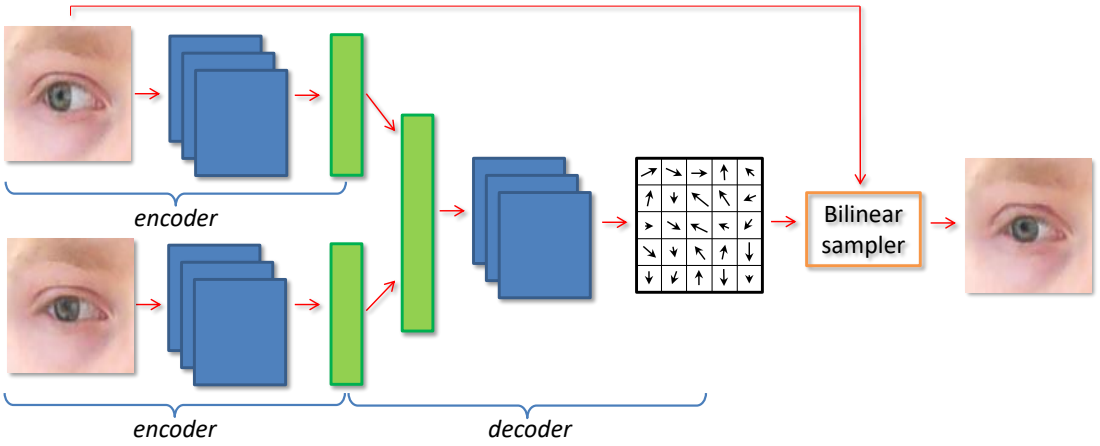


FIGURE 6.3: Architecture of unsupervised gaze redirection training. Two images of the same eye with different gaze direction are passed to the *encoder* network that outputs their latent representations. These representations are then concatenated and passed to the decoder network that outputs the predicted 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 [Jaderberg et al., 2015] 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 [Kingma and Ba, 2014]. 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 $P(\alpha_q) = \{(I_1^1, I_2^1), \ldots, (I_1^n, I_2^n)\}$ 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).

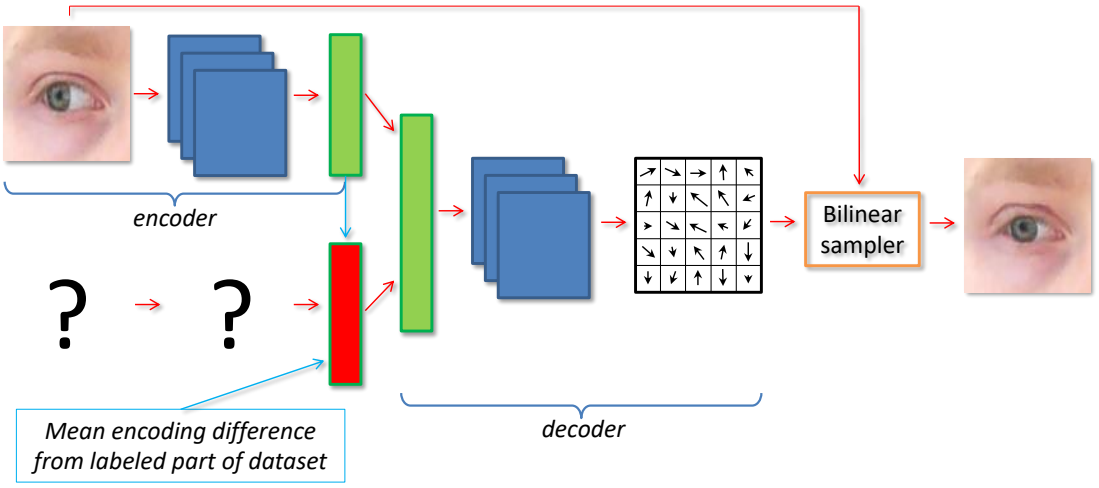


FIGURE 6.4: Image analogy-making in test-time. In test time we do not know second image, we want to produce it. Thus, we do not have its representation for the encoder. The solution is to approximate the unknown representation in the latent space. The estimate of the unknown embedding is the known representation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

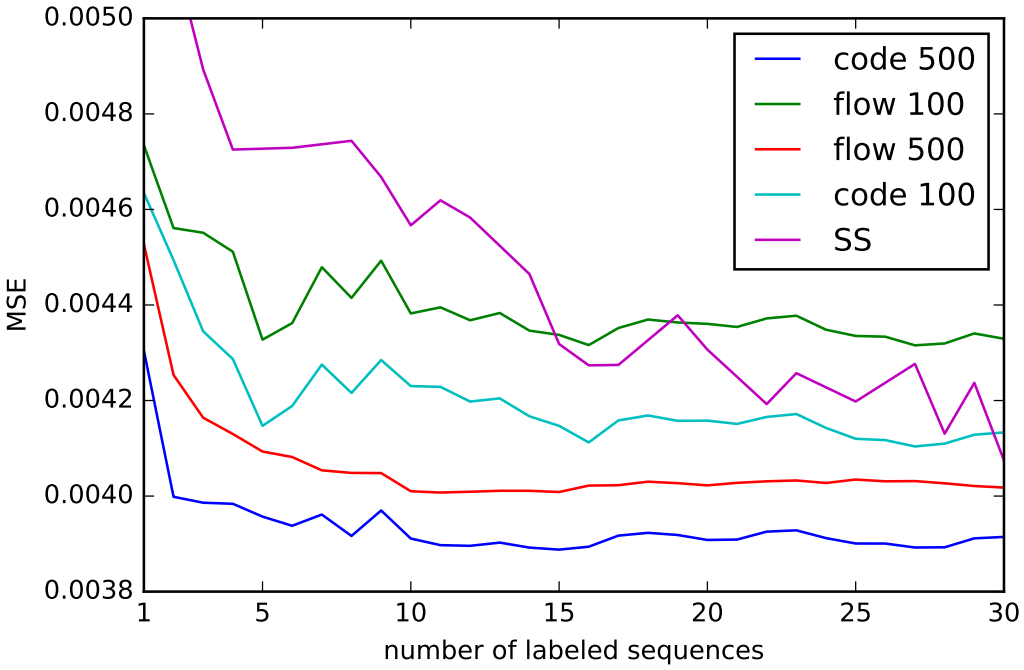


FIGURE 6.6: **Quantitative comparison of methods: errors for redirection on 15 degrees upwards.** Horizontal axis shows the number of sequences labeled with gaze direction provided to the methods. All methods are trained for arbitrary redirection angle, but applied to a testing setting with 15 degrees 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.

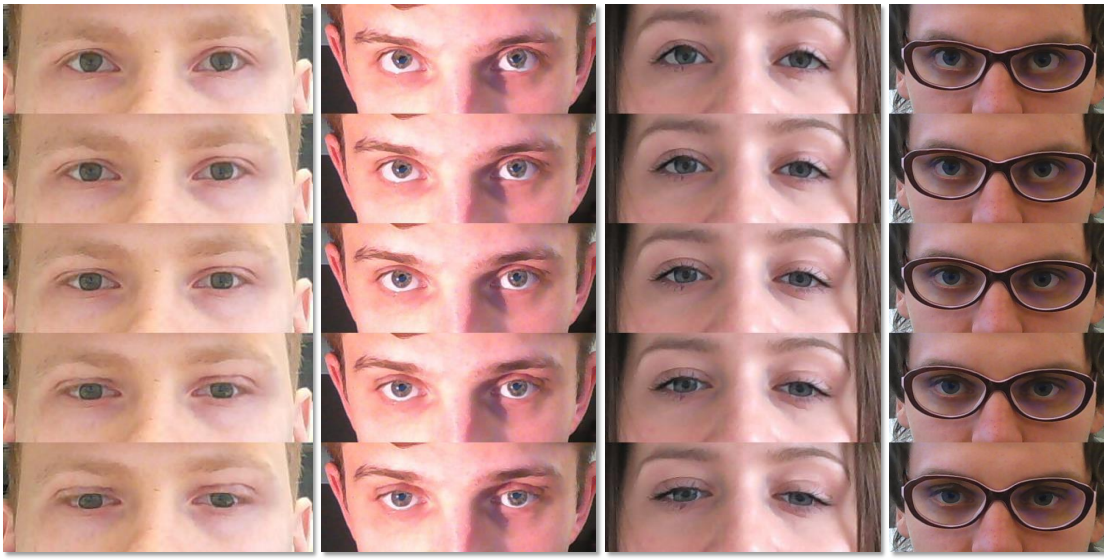


FIGURE 6.8: **Vertical redirection.** The redirection angle ranges from $-15°$ to $15°$ relative to the central input image. In all columns the image in the center row is the input. Convincing continuous gaze redirection is achieved.

Chapter 7

Conclusions and summary

The thesis presents the warping-based approach to image re-synthesis with the target transformation defined by a dataset of input-output examples and examines its application to the gaze redirection task. The warping-based approach enhances the photorealism of the results as compared to direct regression of output pixels which has difficulty synthesizing high-resolution images. [Seitz and Dyer, 1997] and similar works suggest constructing the warping field based on geometrical projection properties, but to the best of my knowledge, this thesis is the first to suggest learning the warping field from the dataset of examples [Kononenko and Lempitsky, 2015].

Four methods are proposed for learning a warping field predictor. Below I will once again summarize the four methods and demonstrate their novelty with reference to the majority of relevant prior publications on the topic. Three of the four methods learn to predict the warping field in a weakly-supervised setting. The first system (Chapter 3, [Kononenko and Lempitsky, 2015]) is based on a special kind of weakly-supervised random forest with a structured output. Its novelty with respect to an earlier study [Fanello et al., 2014] is that both the optimal warping vector and the distribution of the replacement error depending on the warping vector are stored in the leaves of the tree. Storing these distributions also facilitates more robust tree ensembling by summing up distributions coming from different trees. Furthermore, the split evaluation criterion is new and specific to the proposed type of random forest. The second system (Chapter 4, [Ganin et al., 2016])predicts the warping field using a deep convolutional network with a coarse-to-fine architecture of warping modules and embeds the desired redirection angle and feature points as image-sized maps. Along with warping, the lightness correction module is a strong contributor to photorealism enhancement. The third system (Chapter 5, [Kononenko et al., 2017]) ccan be regarded as a hybrid of the first two, as

it essentially condenses the neural network into a random forest, while boasting high-quality results, fast operation, and very compact models. The novelty of the proposed architecture as compared to earlier studies on the teacher-student architectures [Romero et al., 2015, Bucilu et al., 2006, Hinton et al., 2015] lies in its specific teacher-student combination where the teacher is a weakly-supervised neural network and the student is a random forest. Such an arrangement allows integrating the representation power of the deep neural network in the fast forest model. The fourth system (Chapter 6) driven by the labeled database collection challenge operates in a semi-supervised scenario that requires a small labeled part of the dataset only. It focuses on learning to embed images into a low-dimensional latent space and making analogies in that space. The novelty of the method as compared to [Reed et al., 2015] is that it uses a combination of unsupervised training on pairs of images with no gaze labels and no requirement for analogy-forming quadruplets, on the one hand, and image analogy making at test time, on the other hand. Another distinctive feature of the method is that the analogy making result comes in the form of a warping field rather than a re-synthesized image itself. This, once again, boosts the photorealism of the result.

As applied to gaze redirection, the approach is common to all the methods and consists in re-synthesizing the eye region in order to emulate the change in gaze direction without changing the head pose. The two forest-based systems redirect the gaze by a fixed angle and run in real time on a single CPU core. The Deep Warp system takes the redirection angle as an input and thus allows changing the gaze direction continuously within a certain range, while also producing a higher quality result. The semi-supervised system also redirects the 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 several limitations in common. The initial idea of re-synthesizing the eye vicinity could fail for very large angles, such as 90°, simply because any attempt to change the gaze direction by this angle without turning the head would look unnatural. However, for small angles, at least in the range ±30°, the results look photorealistic provided the eye regions have been synthesized naturally.

With a large database collected, machine learning can be used to learn target transformation a concept that was not contemplated in any papers on gaze correction published prior to this work. Direct re-synthesis versus warping based approach was one of the crucial choices in all the models. The first approach involves direct regression of the target image pixels. Applied in the experiments with gaze correction data, this approach resulted in noticeable blurriness and loss of fine details. These conclusions correlate well with the results reported in [Kulkarni et al., 2015, Ghodrati et al., 2016, Reed et al., 2015, Goodfellow et al., 2014]. Generally, warping-based re-synthesis produces sharp images but has certain limitations in synthesizing realistic disoccluded areas. The key insight of this thesis is that the 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 direct re-synthesis in the gaze redirection application. The disoccluded 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 [Park et al., 2017].

The two types of predictors used in this work are random forests and convolutional neural networks. In general, neural networks yield better results, both quantitatively and qualitatively, due to their representation power. This is in line 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 of a random forest architecture by a neural network allows attaining the same speed and nearly the same quality as in the network-based model. As regards gaze correction in videoconferencing, I consider this solution to be the best choice among the different solutions proposed in this work.

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

The weak point of the forest-based methods as they are implemented now is the fixed redirection angle. The neural network models, by contrast, take the redirection angle as an additional input and are thus capable of producing images with different gaze directions. The limitation of forest-based models is not critical for the videoconference application where a single correction angle making up 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 the person to the screen can be a solution too, if flexibility in the redirection angle is needed. This would be much easier to implement in real time on a consumer device without a GPU than to design 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 the gaze by an arbitrary angle. On the other hand, such applications have less stringent requirements in terms of running speed, so the Deep Warp system based on deep neural networks is a preferred solution in their case.

The models mentioned above require a dataset with fully labeled gaze redirection. Since such a dataset is difficult to collect, the semi-supervised approach for training a neural network model was suggested. With a large amount of unlabeled data available, the semi-supervised system shows significant improvement over purely supervised models that ignore eye images with unknown gaze direction. Data labelling is often the most expensive operation in computer vision applications. In the particular case of data collection for this study, the requirement for labeled data was a major hindrance. Thus switching to the semi-supervised approach appears as a useful opportunity in handling large datasets.

Finally, a well-grounded conclusion about the perceptual quality of the results cannot be made based on the reconstruction error alone and should be backed by a user study in problems like gaze redirection. Some methods that show slightly worse results in terms of the ℓ_2 reconstruction error are still perceptually good enough for users. The user study reveals high photorealism of the suggested methods, with the guess ratio approaching a random guess. In general, the user studies confirm that the proposed methods can deliver fairly realistic results despite the very high standards imposed by the human visual perception system as regards the realism of face and eye images. The proposed methods have been recognized both by the academic community, as evidenced by several publications, and by the industry, with the commercialization process under way and the license purchased by a major company.

Bibliography

[Amit and Geman, 1997] Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588.

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

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

[Baltrusaitis et al., 2013] Baltrusaitis, T., Robinson, P., and Morency, L.-P. (2013). 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.

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

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

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

[Brock et al., 2017] Brock, A., Lim, T., Ritchie, J. M., and Weston, N. (2017). Neural photo editing with introspective adversarial networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

[Bucilu et al., 2006] Bucilu, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

[Cham et al., 2002] Cham, T., Krishnamoorthy, S., and Jones, M. (2002). 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.

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

[Criminisi and Shotton, 2013] Criminisi, A. and Shotton, J. (2013). *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media.

[Criminisi et al., 2003] Criminisi, A., Shotton, J., Blake, A., and Torr, P. H. (2003). Gaze manipulation for one-to-one teleconferencing. In *IEEE International Conference on Computer Vision (ICCV)*, pages 191–198.

[Denton et al., 2015] Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494.

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

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

[Dosovitskiy et al., 2015] Dosovitskiy, A., Tobias Springenberg, J., and Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *CVPR*.

[Everingham et al., 2006] Everingham, M., Sivic, J., and Zisserman, A. (2006). Hello! my name is... buffy–automatic naming of characters in tv video. In *BMVC*.

[Fanelli et al., 2013] Fanelli, G., Dantone, M., Gall, J., Fossati, A., and Gool, L. J. V. (2013). Random forests for real time 3d face analysis. *International Journal of Computer Vision*, 101(3):437–458.

[Fanello et al., 2014] Fanello, S. R., Keskin, C., Kohli, P., Izadi, S., Shotton, J., Criminisi, A., Pattacini, U., and Paek, T. (2014). Filter forests for learning data-dependent convolutional kernels. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1709–1716.

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

[Ganin et al., 2016] Ganin, Y., Kononenko, D., Sungatullina, D., and Lempitsky, V. (2016). Deepwarp: Photorealistic image resynthesis for gaze manipulation. In *European Conference on Computer Vision*, pages 311–326. Springer.

[Gatys et al., 2015] Gatys, L., Ecker, A. S., and Bethge, M. (2015). Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270.

[Ghodrati et al., 2016] Ghodrati, A., Jia, X., Pedersoli, M., and Tuytelaars, T. (2016). Towards automatic image editing: Learning to see another you. In *Proceedings BMVC 2016*, pages 1–11.

[Giger et al., 2014] Giger, D., Bazin, J.-C., Kuster, C., Popa, T., and Gross, M. (2014). Gaze correction with a single webcam. In *IEEE International Conference on Multimedia & Expo*.

[Glassner, 2014] Glassner, A. S. (2014). *Principles of Digital Image Synthesis*. Morgan Kaufmann.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[Gregor et al., 2015] Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1462–1471.

[Guinnip et al., 2004] Guinnip, D. T., Lai, S., and Yang, R. (2004). View-dependent textured splatting for rendering live scenes. In *ACM SIGGRAPH 2004 Posters*, page 51. ACM.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

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

[Hertzmann et al., 2001] Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., and Salesin, D. H. (2001). Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *stat*, 1050:9.

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

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Jaderberg et al., 2015] Jaderberg, M., Simonyan, K., and Zisserman, A. (2015). Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025.

[Johnson et al., 2016] Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer.

[Jones et al., 2009] Jones, A., Lang, M., Fyffe, G., Yu, X., Busch, J., McDowall, I., Bolas, M. T., and Debevec, P. E. (2009). Achieving eye contact in a one-to-many 3D video teleconferencing system. *ACM Trans. Graph.*, 28(3).

[Kazemi and Sullivan, 2014] Kazemi, V. and Sullivan, J. (2014). 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.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*.

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

[Kononenko, 2015] Kononenko, D. (2015). Project webpage machine learning-based gaze correction. http://tinyurl.com/gazecorr. [Online; accessed 17-Aug-2017].

[Kononenko, 2017] Kononenko, D. (2017). Gaze correction for videoconferencing. https://youtu.be/sw31vBxQUNs. [Online; accessed 17-Aug-2017].

[Kononenko et al., 2017] Kononenko, D., Ganin, Y., Sungatullina, D., and Lempitsky, V. (2017). Photorealistic monocular gaze redirection using machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Kononenko and Lempitsky, 2015] Kononenko, D. and 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*, pages 4667–4675.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Kulkarni et al., 2015] Kulkarni, T. D., Whitney, W. F., Kohli, P., and Tenenbaum, J. (2015). Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547.

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

[Kuster et al., 2012] Kuster, C., Popa, T., Bazin, J.-C., Gotsman, C., and Gross, M. (2012). Gaze correction for home video conferencing. *ACM Transactions on Graphics (TOG)*, 31(6):174:1–174:6.

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

[LeCun, 1989] LeCun, Y. (1989). Generalization and network design strategies. *Connectionism in perspective*, pages 143–155.

[Ledig et al., 2017] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition, 2017*.

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

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

[Liu et al., 2015] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738.

[Liu et al., 2017] Liu, Z., Yeh, R., Tang, X., Liu, Y., and Agarwala, A. (2017). Video frame synthesis using deep voxel flow. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[Liu et al., 2001] Liu, Z., Zhang, Z., Jacobs, C., and Cohen, M. (2001). Rapid modeling of animated faces from video. *Computer Animation and Virtual Worlds*, 12(4):227–240.

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

[Mahendran and Vedaldi, 2015] Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *CVPR*.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Mirza and Osindero, 2014] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

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

[Okada et al., 1994] Okada, K.-I., Maeda, F., Ichikawaa, Y., and Matsushita, Y. (1994). 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.

[Park et al., 2017] Park, E., Yang, J., Yumer, E., Ceylan, D., and Berg, A. C. (2017). Transformation-grounded image generation network for novel 3d view synthesis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

[Qin et al., 2015] Qin, Y., Lien, K.-C., Turk, M., and Höllerer, T. (2015). Eye gaze correction with a single webcam based on eye-replacement. In *International Symposium on Visual Computing*, pages 599–609. Springer.

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

[RealD, nd] RealD (n.d.). Reald website. https://www.reald.com. Online; accessed 23-Aug-2017.

[Reed et al., 2015] Reed, S. E., Zhang, Y., Zhang, Y., and Lee, H. (2015). Deep visual analogy-making. In *Advances in neural information processing systems*, pages 1252–1260.

[Ren et al., 2014] Ren, S., Cao, X., Wei, Y., and Sun, J. (2014). 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.

[Rezende et al., 2014] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1278–1286.

[Romero et al., 2015] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

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

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

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

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE.

[Shotton et al., 2013] Shotton, J., Girshick, R. B., Fitzgibbon, A. W., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., and Blake, A. (2013). Efficient human pose estimation from single depth images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2821–2840.

[Shu et al., 2016] Shu, Z., Shechtman, E., Samaras, D., and Hadap, S. (2016). Eyeopener: Editing eyes in the wild. *ACM Transactions on Graphics*, 36(1).

[Shu et al., 2017] Shu, Z., Yumer, E., Hadap, S., Sunkavalli, K., Shechtman, E., and Samaras, D. (2017). Neural face editing with intrinsic image disentangling. In *IEEE Conference on Computer Vision and Pattern Recognition, 2017*.

[Smith et al., 2013] Smith, B. A., Yin, Q., Feiner, S. K., and Nayar, S. K. (2013). 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.

[Soomro et al., 2012] Soomro, K., Zamir, A. R., and Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402.

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

[Upchurch et al., 2017] Upchurch, P., Gardner, J., Bala, K., Pless, R., Snavely, N., and Weinberger, K. (2017). Deep feature interpolation for image content changes. In *IEEE Conference on Computer Vision and Pattern Recognition, 2017*.

[Wallis et al., 2015] Wallis, L. J., Range, F., Müller, C. A., Serisier, S., Huber, L., and Virányi, Z. (2015). Training for eye contact modulates gaze following in dogs. *Animal behaviour*, 106:27–35.

[Wheeler et al., 1979] Wheeler, R. W., Baron, J. C., Michell, S., and Ginsburg, H. J. (1979). Eye contact and the perception of intelligence. *Bulletin of the Psychonomic Society*, 13(2):101–102.

[Wikipedia, nda] Wikipedia (n.d.a). Bilinear interpolation — Wikipedia, the free encyclopedia. [Online; accessed 21-September-2017].

[Wikipedia, ndb] Wikipedia (n.d.b). Uncanny valley — Wikipedia, the free encyclopedia. [Online; accessed 21-September-2017].

[Wolberg, 1990] Wolberg, G. (1990). *Digital image warping*, volume 10662. IEEE computer society press Los Alamitos, CA.

[Wolf et al., 2010] Wolf, L., Freund, Z., and Avidan, S. (2010). An eye for an eye: A single camera gaze-replacement method. In *Computer Vision and Pattern Recognition (CVPR)*, pages 817–824.

[Wood et al., 2017] Wood, E., Baltrusaitis, T., Morency, L., Robinson, P., and Bulling, A. (2017). Gazedirector: Fully articulated eye gaze redirection in video. *CoRR*, abs/1704.08763.

[Wood et al., 2016] Wood, E., Baltrušaitis, T., Morency, L.-P., Robinson, P., and Bulling, A. (2016). A 3d morphable eye region model for gaze estimation. In *European Conference on Computer Vision*, pages 297–313. Springer.

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

[Xue et al., 2016] Xue, T., Wu, J., Bouman, K., and Freeman, B. (2016). Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, pages 91–99.

[Yang et al., 2011] Yang, F., Wang, J., Shechtman, E., Bourdev, L., and Metaxas, D. (2011). Expression flow for 3d-aware face component transfer. In *ACM Transactions on Graphics (TOG)*, volume 30, page 60. ACM.

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

[Yeh et al., 2016] Yeh, R., Liu, Z., Goldman, D. B., and Agarwala, A. (2016). Semantic facial expression editing using autoencoded flow. *CoRR*, abs/1611.09961.

[Yin et al., 2007] Yin, P., Criminisi, A., Winn, J. M., and Essa, I. A. (2007). Tree-based classifiers for bilayer video segmentation. In *Computer Vision and Pattern Recognition (CVPR)*.

[Yip and Jin, 2003] Yip, B. and Jin, J. S. (2003). 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.

[Yuille et al., 1992] Yuille, A. L., Hallinan, P. W., and Cohen, D. S. (1992). Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2):99–111.

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer.

[Zhou et al., 2016] Zhou, T., Tulsiani, S., Sun, W., Malik, J., and Efros, A. A. (2016). View synthesis by appearance flow. In *European Conference on Computer Vision*, pages 286–301. Springer.

[Zhu et al., 2009] Zhu, J., Liao, M., Yang, R., and Pan, Z. (2009). 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.

[Zhu et al., 2008] Zhu, J., Wang, L., Yang, R., and Davis, J. (2008). 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.

[Zhu et al., 2011] Zhu, J., Yang, R., and Xiang, X. (2011). Eye contact in video conference via fusion of time-of-flight depth sensor and stereo. *3D Research*, 2(3):1–10.

[Zhu et al., 2016] Zhu, J.-Y., Krähenbühl, P., Shechtman, E., and Efros, A. A. (2016). Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*, pages 597–613. Springer.