Photorealistic Monocular Gaze Redirection Using Machine Learning

Daniil Kononenko*, Yaroslav Ganin†, Diana Sungatullina*,Victor Lempitsky*
*Skolkovo Institute of Science and Technology
{daniil.kononenko,d.sungatullina,lempitsky}@skoltech.ru
†Université de Montréal
yaroslav.ganin@gmail.com

Abstract—We propose a general approach to the gaze redirection problem in images that utilizes machine learning. The idea is to learn to re-synthesize images by training on pairs of images with known disparities between gaze directions. We show that 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.
We describe and compare three instantiations of our idea. 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), being particularly suitable for the videoconferencing gaze correction. The second system is based on a deep architecture and allows 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 a teacher network in our experiments, and thus provides a highly realistic real-time monocular solution to the gaze correction problem. We present in-depth assessment and comparisons of the proposed systems based on quantitative measurements and a user study.

Index Terms—Gaze redirection, machine learning, deep learning, random forest, weakly-supervised learning, image resynthesis.

1 Introduction

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 [1]) can infer a lot of information about the owner of the eyes, her intent, her mood, and the world around her, from the appearance of the eyes and, in particular, from the direction of the gaze. Overall, the role of gaze in human communication is long known to be very high [2].

The task of *gaze redirection* (i.e. image modification in order to make an impression of changed gaze direction) emerges in several scenarios. First and foremost, there is a problem of gaze in videoconferencing that has been attracting researchers and engineers for a long time. The problem manifests itself as the inability of the people engaged into a videoconferencing to maintain gaze contact. The lack of gaze contact is due to the disparity between the image of the interlocutor’s face on the screen and the camera (as a result, while the gaze into the eyes of the other person on the screen is perceived as a downward stare by the interlocutor). Another common scenario that needs gaze redirection are “talking head”-type videos with teleprompting. Yet another example is photo editing and movie post-production in order to make gaze direction consistent with the ideas of the photographer or the movie director.

All of these scenarios put very high demands on the realism of the result of the digital alteration, and some of them also require real-time or near real-time operation. To meet these challenges, we propose three systems for monocular gaze correction. All systems are based on supervised machine learning, for which a large number of image pairs describing the gaze redirection process is utilized. We use a specially collected dataset that depicts the change of the appearance under gaze redirection in real life.

For an input image frame, most previous systems for gaze correction synthesize a novel view of a scene from a virtual viewpoint co-located with the screen [3], [4], [5], [6]. Alternatively, a virtual view restricted to the face region is synthesized and stitched into the original video stream [7], [8]. Novel view synthesis is however a challenging task, even in constrained conditions, due to such effects as (dis)-occlusion and geometry estimation uncertainties. Stitching real and synthetic views can alleviate some of these problems, but it often leads to distortions through the multi-perspective nature of the stitched images.

We do not attempt to synthesize a view for a virtual camera. Instead, our method emulates the change in the appearance resulting from a person changing her gaze direction by a certain angle (e.g. ten degrees upwards), while keeping the head pose unchanged (Figure 1). Emulating such gaze redirection is still challenging, as it is associated with (a) complex non-rigid motion of eye muscles, eyelids, and eyeballs, (b) complex occlusion/dis-occlusion of the eyeballs by the eyelids, (c) change in illumination patterns due to the complex changes in normal orientation.

Our 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. We use supervised learning, as the training proceeds by observing multiple pairs of images with altered gaze direction at training time. The main challenge of such approach is to devise learning methods that can generalize to people and imaging conditions unseen during training, while also meeting the requirements of realism and sufficient speed at test-time. Towards this end we present three systems that achieve these goals in varying degrees.

The first of our systems is based on a special kind of randomized decision tree ensembles called *eye flow forests* that are learned in a weakly-supervised manner. At training time, this system observes pairs of images, where each pair contains the face of

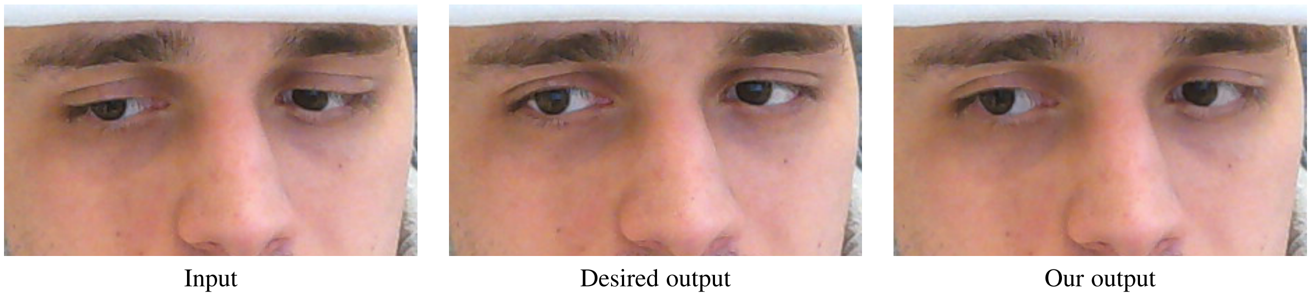


Fig. 1. The setting for monocular gaze redirection. Left – an input frame with the gaze directed below the camera. Middle – a "ground truth" frame with the gaze directed 15 degrees higher than in the input. Given an input image and the desired change in angle and direction ("15 degrees higher") our method 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 (Section 3.4) is shown on the right. In this particular example, the computation time of the method is 5 ms on a single laptop core (excluding feature point localization). Such speed makes our system suitable for real-time use in videoconferencing.

the same person with a fixed angular difference in gaze direction. It then learns to synthesize the second image of a pair from the first one by predicting a warping flow field. After learning, the system gets the ability to redirect the gaze of a previously unseen person by the same angular difference as in the training set. The system synthesizes realistic views with a gaze systematically redirected by $10 - 30$ degrees in our experiments. At test-time, the system accomplishes gaze correction using simple pixel replacement operations that are localized to the vicinity of persons eyes, thus achieving high computational efficiency. Our implementation runs in real-time on a single core of a laptop. Although the results are of a high perceptual quality (see Section 4.3), the system still leaves some room for improvement and artifact reduction. Another, less critical deficiency of the system is a relatively large memory footprint of the learned models.

Our second system is based on a deep feed-forward architecture that combines several principles of operation (coarse-to-fine processing, image warping, intensity multiplication). The architecture is trained end-to-end in a supervised way using a specially collected dataset that depicts the change of the appearance under gaze redirection by different angles. Qualitative and quantitative evaluation demonstrate that our 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 for the first system, but the efficiency falls short of real-time CPU operation, making the method impractical for video-conferencing using most consumer devices. Such approach is still practical for image and video-editing application scenarios outlined above. Our system also contributes to an actively-developing field of image generation with deep models.

Our third system 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, we use an output of the second system, effectively making our deep architecture to "teach" the random forest. At training time, the new system observes images and the corresponding pixel 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 redirects gaze by applying the flow predicted by the random forest to the input image. As shown in our experiments, the trained forest manages to approach the quality of the teacher network, outperforming the quality of 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, resulting models have much smaller memory footprint in comparison to the weakly supervised forest.

The rest of the paper is organized as follows. We review the related work on gaze correction, randomized trees, and image re-synthesis using deep learning (Section 2). We introduce our general approach and the three proposed systems in Section 3. In Section 4, we perform qualitative and quantitative validation of the proposed systems. Finally, we draw some conclusions in Section 5.

2 Related work

Gaze correction in videoconferencing. 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 [9], [10]. Another group of methods offers a mixed software/hardware solution and proceeds in two steps. First, a dense depth map is estimated either through the use of stereomatching [3], [4] or using RGB-D cameras [7], [11]. Then, a new synthetic view corresponding to a virtual camera located behind the screen is created in real-time. A common problem with the novel view synthesis is filling dis-occluded regions. Generally, reliance on additional hardware represents an obvious obstacle to the wide adaptation of these techniques.

While a certain number of purely software, monocular gaze correction approaches have been suggested [5], [6], most of them have been generally unable to synthesize realistic images while meeting the requirements of realism and being able to alter the perceived gaze direction sufficiently. One exception is the system [12] that first prerecords a sequence of frames where a person gazes into the camera, and then, at the conference time, replaces the eye regions with the eye region taken from one of the prerecorded frames (more recently, [13] suggested a similar approach to replace closed eyes with open eyes for photo editing). The downside of [12], is that while the obtained images achieve sufficient realism, the gaze in the synthesized image remains "locked" staring unnaturally into the camera irrespective of the actual movement of eyes. A related drawback is that the system needs to prerecord a sufficient number of diverse images of the person eyes. Our systems do not suffer from either of these limitations.

The more recent system [8] uses monocular real-time approximate fitting of head model. Similarly to [7], the approximate

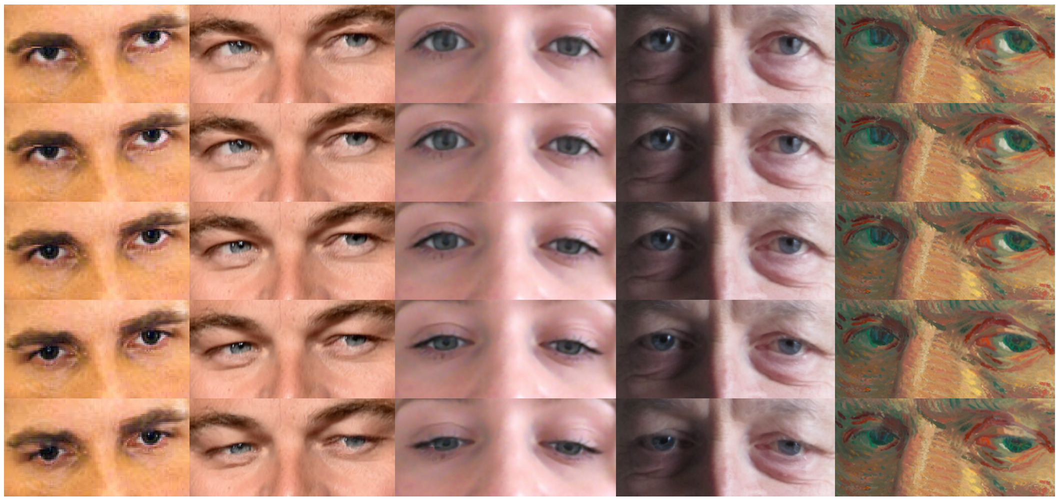


Fig. 2. Gaze redirection with our neural network-based system trained for vertical gaze redirection. The model takes an input image (middle row) and the desired redirection angle (here varying between -15 and +15 degrees) and re-synthesize the new image with the new gaze direction. Note the preservation of fine details including specular highlights in the re-synthesized images.

geometry is used to synthesize a high-quality novel view that is then stitched with the initial view through real-time optimization. The limitations of [8] are the potential face distortion due to multi-perspective nature of the output images, as well as the need for GPU to maintain the real-time operation. Prerecording heavily occluded head parts (under chin) before each video-conference is also needed.

Decision forests in computer vision. The variant of our system based on eye flow trees continues the long line of real-time computer vision systems based on randomized decision trees [14], [15] that includes real-time object pose estimation [16], head pose estimation [17], human body pose estimation [18], background segmentation [19], etc. While classical random forests are trained to do classification or regression, the trees in our method predict some (unnormalized) distributions. Our method is thus related to other methods that use structured-output random forests (e.g. [20], [21]). Finally, the trees in our method are trained under weak supervision, and this relates our work to such methods as [22].

Deep learning and image synthesis. Image synthesis using neural networks is receiving growing attention [23], [24], [25], [26], [27], [28], [29]. More related to our second system are methods that learn to transform input images in certain ways [30], [31], [32]. These methods proceed by learning internal compact representations of images using encoder-decoder (autoencoder) architectures, and then transforming images by changing their internal representation in a certain way that can be trained from examples. We have conducted numerous experiments following this approach combining standard autoencoders with several ideas that have reported to improve the result (convolutional and up-convolutional layers [24], [33], adversarial loss [25], variational autoencoders [34]). However, despite our efforts, we have found that for large enough image resolution, the outputs of the network lacked high-frequency details and were biased towards typical mean of the training data ("regression-to-mean" effect). This is consistent with the results demonstrated in [30], [31], [32] that also exhibit noticeable bluring. The recent work [29] addresses this problem using a new cross-conlutional layer that models correlation between motion and image content as well as the uncertainty of the motion field. The network then encodes input image pyramid into multiple feature maps and convolves these maps with different kernels.

Compared to [30], [31], [32], our approach can learn to perform a restricted set of image transformations. However, the perceptual quality and, in particular, the amount of high-frequency details is considerably better in the case of our method due to the fact that we deliberately avoid any input data compression within the processing pipeline. This is crucial for the class of applications that we consider.

The idea of spatial warping that lies in the core of the proposed system has been previously suggested in [35]. In relation to [35], parts of our architecture can be seen as spatial transformers with the localization network directly predicting a sampling grid instead of low-dimensional transformation parameters.

Finally, in parallel with our work, several papers presented a related approach with a deep network that predicts warping fields (flow) for image editing. The network architecture as well as classes of objects and scenes in [36] differ from ours. Their architecture also lacks an equivalent of our lightness correction module. The work [37] suggests synthesize flow fields that manipulate facial expression. Unlike us, they used VAE-based architecture with regularization on the latent space representation. In [38] the idea of warping flow is used for predicting the missed frame in a video sequence. In [39], the warping flow is used to fill in disoccluded pixels, based on the prediction of the visibility map.

Teacher-student architectures. The idea to use output of very precise but large and slow model as a supervision for a faster architecture goes back to at least [40]. More recently [41] suggested to train a "student" network, from the softened output of an ensemble of wider networks ("teacher" networks), allowing the student network to capture not only the information provided by the true labels, but also the finer structure learned by the teacher network. This idea was further developed in [42] that uses activations of several hidden layers of the teacher network as a guidance for the student network.

Conference versions. The first and the second systems (as well as our general approach to gaze redirection) were presented in the conference papers [43], [44]. This paper adds the third system, which has the highest practical importance as it in many ways combines the speed of the first system and the quality of the second system. We also add several important details and illustrations, as well as a unified treatment and validation for all three systems, including a new user study that required users to assess the realism of face images with redirected gaze (different from the user study in [44], which showed the users separate eye images).

3 Methods

All our systems are based on supervised learning, as the training proceeds by observing multiple pairs of images with altered gaze direction by a known angle at training time. As was mentioned above, the main challenge of such approach is to devise learning methods that can generalize to people and imaging conditions unseen during training, and all three systems that we discuss below achieve such generalization by learning to predict the warping field (*eye flow*) rather than the output image directly. The second system corrects the warped output with per-pixel brightness modification (while still trying to achieve the redirection mostly by warping).

3.1 General setting

All three systems start by localizing the eye regions and then achieving redirection by localized processing. We now discuss the common elements of the three systems.

3.1.1 Eye localization

The eye localization step within our system is standard, as we use an off-the-shelf real-time face alignment library (e.g. [45], [46]) to localize facial feature points. As the gaze-related appearance change is essentially local to the eye regions, all further operations are performed in the two areas surrounding the two eyes.

For each eye, we thus focus on the feature points $f_1 = (x_1, y_1), f_2 = (x_2, y_2) \dots f_N = (x_N, y_N)$ corresponding to that eye (in the case of [45] there are $N = 7$ feature points). We compute a tight axis-aligned bounding box $\mathcal{B}'$ of those points. After this, we define the final bounding box $\mathcal{B}$ having the same center as $\mathcal{B}'$ and having the width W and height H that are proportional to some characteristic radius Δ (i.e. $W = \alpha\Delta$, $H = \beta\Delta$ for certain constants α, β). The bounding box $\mathcal{B}$ is thus covariant with the scale and the location of the eye, and has a fixed aspect ratio $\alpha : \beta$. We define Δ as the width of the tight bounding box, which equals the distance between the corners of an eye: $\Delta = ||f_1 - f_4||$.

3.1.2 Training dataset

As our approach is based on supervised learning, we have collected a Skoltech Dataset of videos of around 150 people (Figure 3). During recording, to minimize head movement, a person places her head on a special stand and follows with her gaze a moving point on the screen in front of the stand. The sequence of frames synchronized with the point position, from which we can deduce the gaze direction, is recorded using a webcam mounted in the middle of the screen.

About 200 frames for one video sequence are recorded. The angular range is 36° in vertical direction and 60° in horizontal direction. We manually exclude bad shots, where a person is blinking, not changing gaze direction monotonically as anticipated or moving head. For each person we record $2-10$ sequences (450 sequences total), changing the head pose and lighting conditions between different sequences. From each sequence, one can draw about $50 - 80$ *training pairs* for a certain angular difference in gaze directions. Each training pair can be regarded as a training example for supervised learning.

Training samples are cropped using the eye localization procedure described above. We incorporate left and right eyes into one dataset, mirroring right eyes. At test time, we use the same predictor for left and right eyes, mirroring the results.

3.1.3 Redirection by pixel wise replacement

After the eye bounding box is localized, the method needs to alter pixels inside the box to emulate gaze redirection. As mentioned above, we rely on machine learning to accomplish this. 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:

$$O(x,y) = I\left(x + u(x,y), y + v(x,y)\right) \ . \qquad (1)$$

In other words, the pixel value at (x,y) is "copy-pasted" from another location determined by the *eye flow* vector (u,v). Such restriction on the transformation introduces natural regularization in our method. Such approach ensures that the pixels of the output are copied from the input rather than "invented". However, this approach makes the learning problem weakly-supervised, because we do not have flow vectors $(u(x,y), v(x,y))$ for the training pairs, so the learning method should be developed to handle such weak supervision.

3.1.4 Image-independent flow field

Under our approach, we can propose a very simple baseline that suggests a fixed eye flow vector in (1) 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 Eye flow forest

We now describe the three systems sequentially, starting with the system based on *weakly-supervised* random forests (*eye flow forests*). 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 eye 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 (*eye flow trees*). When a pixel (x,y) is passed through an eye 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:

$$I(x + dx, y + dy)[c] - I(x,y)[c] \gtrless \tau \qquad (2)$$

A test of the second kind (*a location test*) is determined by the number of the feature point $i \in \{1, \dots N\}$ and a threshold τ and compares either $x - f_i$ or $y - g_i$ with τ:

$$x - f_i \gtrless \tau \qquad y - g_i \gtrless \tau \qquad (3)$$

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

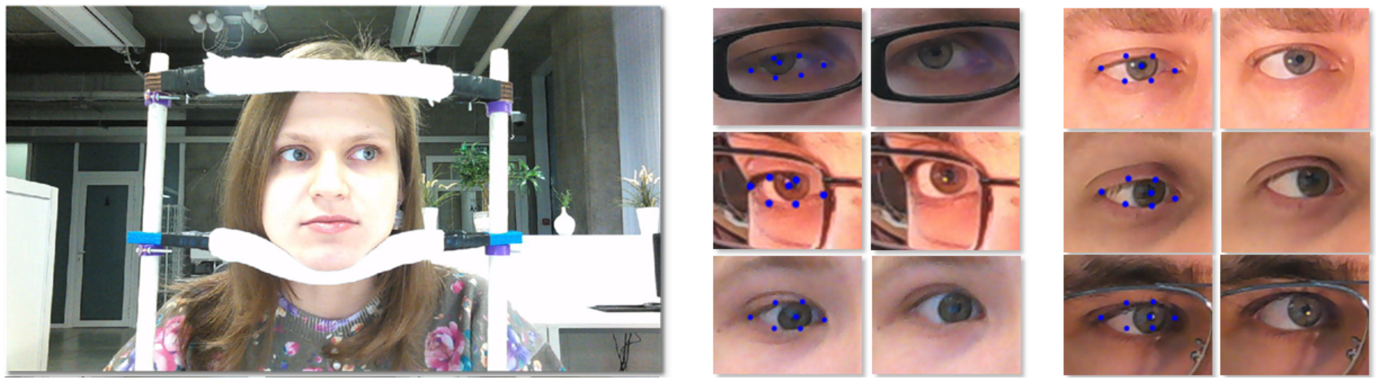


Fig. 3. Left – dataset collection process. Right – examples of training pairs for 15° vertical redirection.

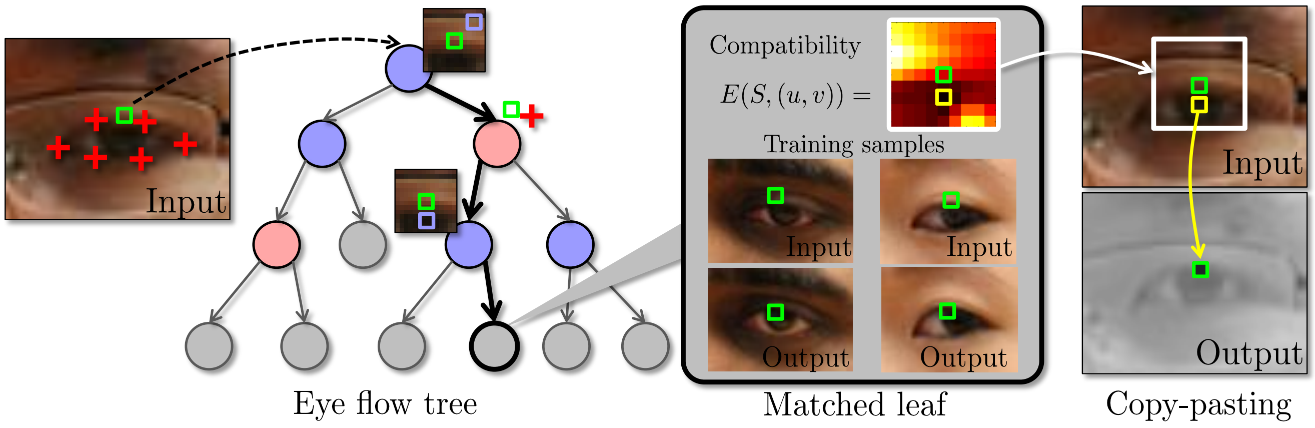


Fig. 4. **Processing of a pixel (green square) at test time in an eye flow tree.** The pixel is passed through an eye flow tree by applying a sequence of tests that compare the position of the pixels w.r.t. the feature points (red crosses) or compare the differences in intensity with adjacent pixels (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 eye flow vectors. The system then takes the optimal eye 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, our actual system would sum up the compatibility scores coming from several trees before making the decision about the eye flow vector to use.

Each of the leaves contain an unnormalized distribution of compatibility score (4) over the eye flow vectors (u, v) for the training examples that fell into that leaf at learning stage. We 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 (1).

Handling scale variations. To make matching and replacement operations covariant with the changes of scale, a special care has to be taken. For this, we 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 we work at the native resolution of the input image. However, when descending an eye flow tree, we multiply the displacements (dx, dy) in (2) and the τ value in (3) by the ratio Δ/Δ_0. Likewise, during copy-paste operations, we multiply the eye 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.

3.2.1 Learning

We assume that a set of training image pairs (I^j, O^j) is given. We 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 1). We 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, we 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 (1). Therefore, each pixel (x, y) within the bounding box B specifies a training tuple $\mathcal{S} = \{(x, y), I, \{(f_i, g_i)\}, O(x, y)\}$, which includes the position (x, y) of the pixel, the input image I it is sampled from, eye feature points $\{(f_i, g_i)\}$ 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, eye flow trees have to be trained in a weakly-supervised manner. This is because each training sample does not include the target vectors $(u(x, y), v(x, y))$ 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 (1) with the same eye flow vector (u, v) produces good result for all training samples that fall into that region. Given a set of training samples $\mathbf{S} = \{\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^K\}$, we define the *compatibility score* E of this set with the eye flow (u, v) in the following natural way:

$$E\Big(\mathbf{S}, (u,v)\Big) = \tag{4}$$
$$\sum_{k=1}^{K} \sum_{c=R,G,B} \Big| I^k(x^k + u, y^k + v)[c] - O^k(x^k, y^k)[c] \Big| \, .$$

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 (1) produces.

Given the compatibility score (4) we can define the *coherence score* $\tilde{E}$ of a set of training samples $\mathbf{S} = \{\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^K\}$ as:

$$\tilde{E}(\mathbf{S}) = \min_{(u,v)\in\mathcal{Q}} E\left(\mathbf{S}, (u,v)\right) \, , \tag{5}$$

Here, $\mathcal{Q}$ denotes the search range for (u, v), which in our implementation we take 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 eye flow vector $(u, v) \in \mathcal{Q}$, i.e. replacement (1) with this flow vector produces colors that are similar to the desired ones.

The coherence score (5) 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 (2),(3) are sampled. Each test is then applied to all samples in the group, thus splitting $\mathbf{S}$ into two subgroups $\mathbf{S}_1$ and $\mathbf{S}_2$. We then define the quality of the split $(\mathbf{S}_1, \mathbf{S}_2)$ as:

$$F(\mathbf{S}_1, \mathbf{S}_2) = \tilde{E}(\mathbf{S}_1) + \tilde{E}(\mathbf{S}_2) + \lambda \, \big| |\mathbf{S}_1| - |\mathbf{S}_2| \big| \, , \tag{6}$$

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 (5) are all "bad" and becomes relatively less important towards the leaves. After all generated tests are scored using (6), 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 $\tilde{E}(\mathbf{S})$ 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 flow field is much easier than training eye flow trees. For this, we consider all training examples for a given location (x, y) and evaluate the compatibility scores (4) 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. We stress that by predicting the eye flow $\big(u(x,y), v(x,y)\big)$ we 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 4, recovering such dependencies using discriminative learning and exploiting them allows to produce sufficiently realistic emulations of gaze redirection.

3.2.2 *Implementation details*

Learning the forest. When learning each split in a node of a tree, we first draw randomly several tests without specifying thresholds. Namely, for each test we first randomly sample a type of the test, choosing between the appearance test and the location test with equal probability. We then we sample parameters of test uniformly from a certain range. We thus sample dx, dy (from the 9×9 neighborhood) and a channel c for appearance tests (2), or the number of the feature point for location tests (3). We then learn an optimal threshold for each of the drawn test. In more detail, denote as h the left-hand-sides of expressions (2), (3), and $h_1, \ldots, h_K$ — all the data sorted by this expression. We 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 (5) and the quality score (6) as inspired by [21]).

To randomize the trees and speed up training, we learn each tree on random part of the data. Afterwards we "repopulate" each tree using the whole training data, i.e. we 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.

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

Numerical parameters. In the current implementation we resize all cropped eyes to the resolution 50×40. We 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. We stop learning splits and make a new leaf if one of the stopping criteria is satisfied: either coherence score (5) 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.

3.3 The deep warp system

In this subsection, we proceed to the discussion of our second system (*deep warp*). Unlike the eye flow forest system, the deep warp system is trained on pairs of images corresponding to eye appearance before and after the redirection by different angles. The redirection angle serves as an additional input parameter that is provided both during training and at test time.

As in the forest-based approach, the bulk of gaze redirection is accomplished via warping the input image (Figure 5). The task

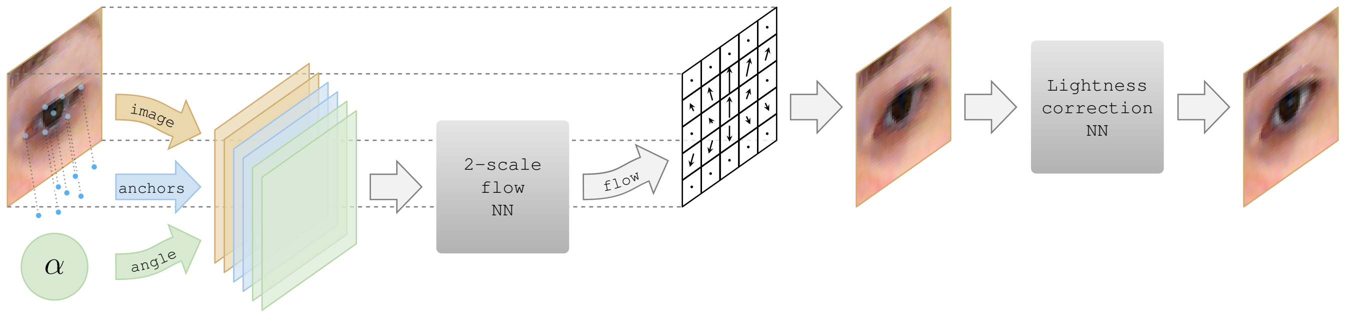
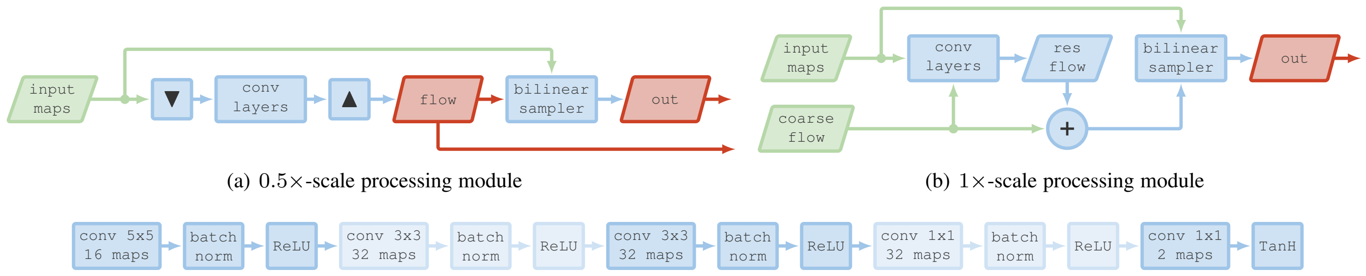


Fig. 5. The deep warp system takes an input eye region, feature points (`anchors`) as well as a correction `angle` α and sends them to the multi-scale neural network (see Section 3.3.1) predicting a `flow` field. The flow field is then applied to the input image to produce an image of a redirected eye. Finally, the output is enhanced by processing with the lightness correction neural network (see Section 3.3.3).



(a) 0.5×-scale processing module

(b) 1×-scale processing module

(c) Convolutional layers

Fig. 6. The architecture of the two warping modules: (`process 0.5×-scale` 6(a) and `process 1×-scale` 6(b)) predicting and applying pixel-flow to the input image; 6(c) represents a fully convolutional sequence of layers inside warping modules.

of the network is therefore the prediction of the warping field. This field is predicted in two stages in a coarse-to-fine manner, where the decisions at the fine scale are being informed by the result of the coarse stage. Beyond coarse-to-fine warping, the photorealism of the result is improved by performing pixel-wise correction of the brightness where the amount of correction is again predicted by the network (Figure 8). All operations outlined above are implemented in a single feed-forward architecture and are trained jointly end-to-end.

3.3.1 Warping modules

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 [47] and ReLU nonlinearities (the actual configuration is shown in Figure 6(c)). To preserve the resolution of the input image, we use ‘same’-mode convolutions (with zero padding), set all strides to one, and avoid using max-pooling.

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. We provide the details below.

Coarse warping. The last convolutional layer of the first (half-scale) warping module (Figure 6(a)) produces a pixel-flow field (a two-channel map), which is then upsampled $\mathbf{D}_{\text{coarse}}(I,\alpha)$ and applied to warp the input image by means of a bilinear sampler $\mathbf{S}$ [35] that finds the *coarse estimate*:

$$O_{\text{coarse}} = \mathbf{S}\left(I, \mathbf{D}_{\text{coarse}}(I,\alpha)\right) \ . \tag{7}$$

Here, the sampling procedure S samples the pixels of O_{coarse} at pixels determined by the flow field:

$$\begin{aligned} &O_{\text{coarse}}(x,y,c) = \\ &\quad I\{x + \mathbf{D}_{\text{coarse}}(I,\alpha)(x,y,1), y + \mathbf{D}_{\text{coarse}}(I,\alpha)(x,y,2), c\}\,, \end{aligned} \tag{8}$$

where c corresponds to a color channel and the curly brackets correspond to bilinear interpolation of $I(\cdot,\cdot,c)$ at a real-valued position. The sampling procedure (7) is piecewise differentiable [35].

Fine warping. In the fine warping module (Figure 6(b)), the rough image estimate O_{coarse} and the upsampled low-resolution flow $\mathbf{D}_{\text{coarse}}(I,\alpha)$ 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 $\mathbf{D}_{\text{res}}$ that amends the half-scale pixel-flow (additively [48]):

$$\begin{aligned} \mathbf{D}(I,\alpha) = \mathbf{D}_{\text{coarse}}(I,\alpha) + \\ \mathbf{D}_{\text{res}}(I,\alpha,O_{\text{coarse}},\mathbf{D}_{\text{coarse}}(I,\alpha))\,, \end{aligned} \tag{9}$$

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

$$O = \mathbf{S}\left(I, \mathbf{D}(I,\alpha)\right) \ . \tag{10}$$

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

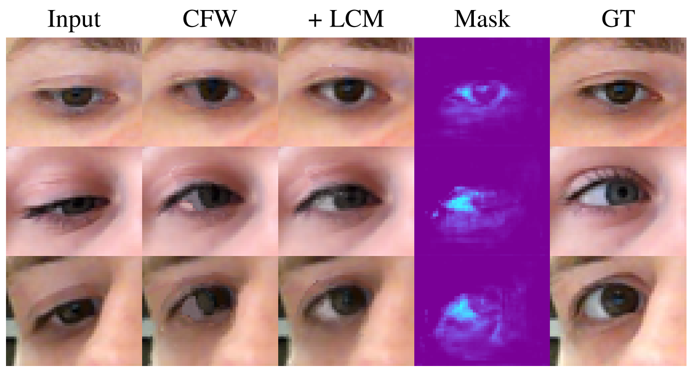
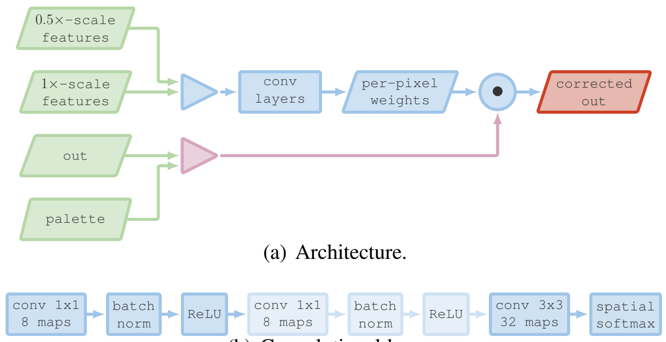


Fig. 7. Visualization of three challenging redirection cases where the **Lightness Correction Module** helps considerably compared to the system based solely on coarse-to-fine warping (CFW), which is having difficulties with expanding the area to the left of the iris. The 'Mask' column shows the soft 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.



(a) Architecture.

(b) Convolutional layers.

Fig. 8. 8(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 8(b)) as input.

of how a redirected eye would look like. This is useful for locating problematic regions which can only be fixed at a finer scale.

3.3.2 *Input encoding*

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 [31], we treat the correction angle as an attribute and embed it into a higher dimensional space using a multilayer perceptron $\mathbf{F}_{\text{angle}}(\alpha)$ with ReLU nonlinearities. The precise architecture is `FC(16)` $\rightarrow$ `ReLU` $\rightarrow$ `FC(16)` $\rightarrow$ `ReLU`. Unlike [31], we do not output separate features 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.

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, we found it 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 (x_i, y_i), we compute a pair of maps:

$$\begin{array}{l}\Delta_x^i[x,y] = x - x_i,\\ \Delta_y^i[x,y] = y - y_i,\end{array} \quad \forall (x,y) \in \{0,\dots,W\} \times \{0,\dots,H\},\tag{11}$$

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).

3.3.3 *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 geometric and require a more general transformation. In addition, the warping approach can struggle to fill in dis-occluded areas in some cases (Figure 7).

To increase the generality of the transformation that can be handled by our deep warp architecture, we add the final lightness adjustment module (Figure 8(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:

$$O_{\text{final}}(x,y,c) = O(x,y,c)\cdot(1 - M(x,y)) + M(x,y),\tag{12}$$

assuming that the brightness in each channel is encoded between zero and one. The resulting pixel colors can thus be regarded as blends between the colors of the warped pixels and the white color. The actual architecture for the lightness correction module in our experiments is shown in Figure 8(b).

3.3.4 *Training procedure*

We used a regular ℓ_2-distance between the synthesized output O_{output} and the ground-truth O_{gt} as the objective function. The model was trained end-to-end on 128-sized batches using Adam optimizer [34]. We found that biasing the selection process for more difficult and unusual head poses and bigger redirection angles improved the results. For this reason, we used the following sampling scheme aimed at reducing the dataset imbalance. We 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 $\geqslant 8°$. This subdivision helps to identify training pairs corresponding to the rare head poses. We form a training batch by picking 4 correction angle bins uniformly at random and sampling 24 "easy" and 8 "hard" examples for each of the chosen bins.

3.3.5 *Implementation details*

The neural network is trained at a fixed spatial scale. At test time, we rescale the input to match the resolution at training time. The predicted flow and lightness correction fields are then bilinearly

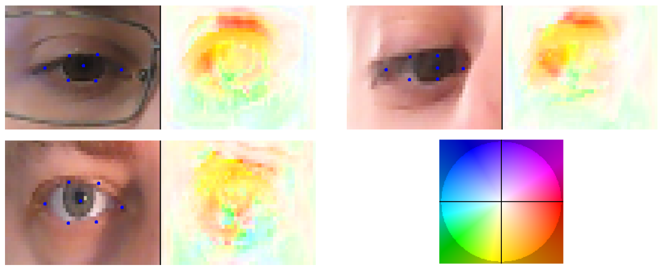


Fig. 9. The output flow of warping modules (Section 3.3.1) on random samples from a training set. The coarse-to-fine model without lightness correction was trained on a task of 15° redirection upwards. This data is used to train a neural network-supervised regression random forest. The right down figure is a color pattern, explaining how the direction of the flow vector is encoded with the color of the pixel. The more intense the pixel is, the longer the flow vector in this pixel is.

rescaled to the original resolution. We also linearly decrease the flow and lightness correction maps to zeros near the border of the cropped eye (four pixels in our experiments). The proposed architecture is implemented in Torch using the bilinear sampler operation provided with [49].

3.4 Neural network-supervised forests

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

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

In more detail, we fix the desired redirection angle and use the sum of the coarse and the fine flow predicted by a deep warp system (the *teacher*) for the given angle as ground-truth data. The examples of such training data are shown in Figure 9.

The input data to the NN-supervised regression forests is the same as to the weakly supervised eye-flow forests, i.e. an image of the eye and its landmark locations. The regression forests apply the same appearance and location tests ((2) and (3)) as the eye-flow forests. As is commonly done for regression forests, we use the split criterion that measures the variance of the two child nodes. In more detail, suppose we have a set of training samples $\mathbf{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K\}$, where each sample is a tuple $\mathcal{S} = \{(x,y), I, \{(f_i, g_i)\}, (dx, dy)\}$ 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 then computed as:

$$Q(\mathbf{S}_1, \mathbf{S}_2) = V(\mathbf{S}_1) + V(\mathbf{S}_2) + \lambda \left| |\mathbf{S}_1| - |\mathbf{S}_2| \right|, \tag{13}$$

where

$$V(\mathbf{S}_i) = V_x(\mathbf{S}_i) + V_y(\mathbf{S}_i) = \sum_{s \in S_i} (dx_s - \overline{dx}_i)^2 + \sum_{s \in S_i} (dy_s - \overline{dy}_i)^2, \tag{14}$$

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

At test time, a pixel is passed down the tree, and the mean 2D flow vector (across the training examples, which fell to the corresponding leaf) is picked. The flows coming from different trees in a forest are averaged to get the final result. As the range of flow is limited, only two bytes are sufficient to store a flow vector in each leaf, which is much smaller compared to weakly supervised forest. The reduction in memory is thus $w^2/2$, where w is the size of the width of the patch in the weakly-supervised eye flow forest system (Figure 4). Moreover, as the supervised task is easier to learn, the depth and the number of leaves can be made lower, resulting in typical memory demand for trained forests of only three megabytes.

The prediction of the lightness correction module 3.3.3 can be incorporated using the second forest, which is applied to the output images after applying the warping predicted by the first NN-supervised forest. The train set for the second forest is the output of the lightness correction module of the neural network and images $\{\hat{I}^j\}$, which are obtained from original set $\{I^j\}$ by applying the first forest and warping procedure (1). The splitting criterion is the same as (13), except that output in this case is one-dimensional vector.

4 Experiments

We now evaluate the performance of the three systems. We show the qualitative results and perform comparisons between the systems (also evaluating the baseline from Section 3.1.4).

4.1 Quantitative evaluation

We evaluate the methods on our dataset. We randomly split the initial set of subjects into the training and the testing sets. We sample image pairs applying the same pre-processing steps as when preparing data for learning (Section 3.1.1).

15° **correction.** In order to have the common ground with the existing systems, we first restrict ourselves to the case of 15° vertical gaze redirection. For this comparison we consider the following models:

1) A system based on weakly supervised eye-flow random forests (*EFF*), described in Section 3.2.
2) A coarse-to-fine warping-based system described in Section 3.3 (*CFW*) without lightness correction trained for 15° vertical gaze redirection.
3) A neural network supervised random forests (*NNSF*), described in Section 3.4, which predicts the output flow of a coarse-to-fine warping-based system without lightness correction. The teacher for the method is *CFW* (the resulting flow on train set).
4) A deep warp coarse-to-fine warping-based system with a lightness correction module trained for 15° vertical gaze redirection (*CFW + LCM*) (Section 3.3.3).
5) A simple baseline - image independent flow field (Section 3.1.4), where the flow is based solely on the relative position of the pixel (*IIF*).

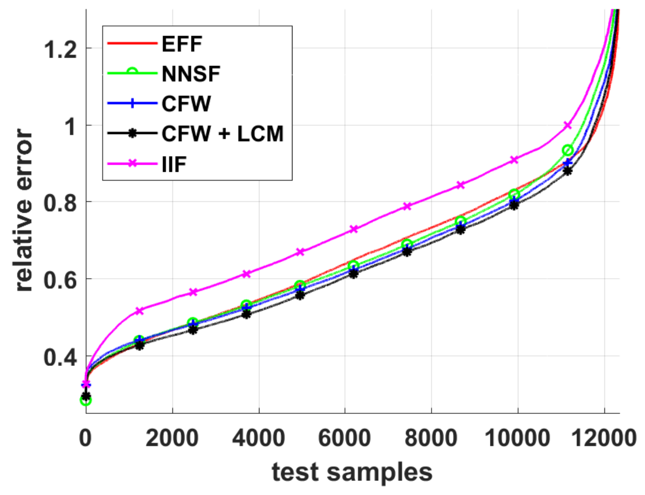


Fig. 10. Ordered errors for 15° vertical gaze redirection (see text for more discussion of the error metric). The best performance is shown by the full coarse-to-fine architecture with the lightness correction module.

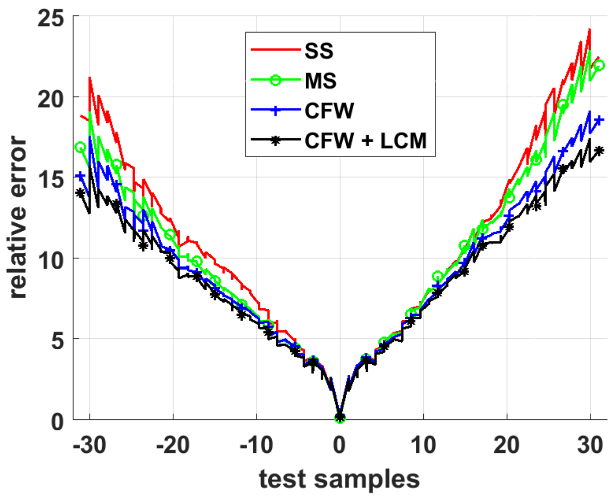


Fig. 11. Distribution of errors over different vertical correction angles. With the increase of the redirection angle, the more comprehensive models show the better quality.

In Figure 10 we present a graph of sorted normalized errors, where all errors are divided by the MSE obtained by an input image taken as an output. For each method, the errors on the test set are sorted. It can be seen, that NN-supervised forest performs comparably to the eye-flow forest. What is more important, this improvement is quite visible in terms of noise and artifacts (Figure 12). 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 3.3.3) is able to show quite significant improvements. Those are mostly cases similar to shown in Figure 7. Unified multiscale models trained to handle different angles (not included in this comparison) are, in general, comparable with the one specialized for 15° redirection. It is also worth noting that even a single-scale model trained for this specific correction angle consistently outperforms eye flow forest, demonstrating the superiority of deep learning.

Arbitrary vertical redirection. We also compare different variants of the deep warp system and plot the error distribution over different redirection angles (Figure 11). The neural network models in this comparison are trained for the task of vertical gaze redirection in the range from -30° to 30°. The following systems are compared to demonstrate the importance of different elements of the deep warp system:

1) A single-scale (*SS*) version of deep warp method. It consists of a single warping module Figure 6(a) operating on the original image scale.
2) A multiscale (*MS*) network without coarse warping. The system processes inputs on two scales and uses features from both scales to predict the final warping transformation.
3) A coarse-to-fine warping-based system described in Section 3.3 (*CFW*) without lightness correction.
4) A coarse-to-fine warping-based system with a lightness correction module (*CFW* + *LCM*) (Section 3.3.3).

For small angles, all systems demonstrate roughly the same performance, but as we increase the amount of correction, the task becomes much harder (which is reflected by the growing error) revealing the difference between the models. The best results are achieved by the full model, which is followed by the multi-scale network without the lightness correction module.

4.2 Qualitative evaluation

To qualitatively compare the systems, we show a *random* subset of our results of redirection by 15 degrees upwards in Figure 12. All methods were trained for 15° vertical redirection. One can observe that the system is able to learn to redirect the gaze of unseen people rather reliably obtaining a close match with the ground truth. Deep warp systems produce the results visually closer to the ground truth, than forest-based systems. The effect of lightness correction is pronounced: on the input image with the invisible sclera in one corner, the system with lightness correction performs clearly better. However, the downside effect of lightness correction could be blurring/lower contrast because of the multiplication procedure (12).

Redirection within range. In Figure 2 and Figure 13, we provide qualitative results of vertical and horizontal gaze redirection in the angular range from -15° to 15° using the model trained for both horizontal and vertical redirection. Some examples showing the limitations of our method are given. The limitations are concerned with cases with severe dis-occlusions, where large areas have to be filled by the network.

Lower resolution images. In Figure 14 we provide results of the NNSF-system for lower input resolutions on randomly sampled images from a test set. For that, we downsample images from the original size of 80×100, to 10×12, 20×25, 40×50 correspondingly. The NNSF-system is then applied to the downsampled versions. The effect of gaze redirection is noticeable even for very low resolutions.

Comparison with Giger et al. In Figure 15 we show the side-by-side comparison with the system [8], which also performs monocular gaze correction for the videoconferencing scenario. The difference in the approaches is clearly visible. While our method redirects gaze and confines the changes to the eye region, while keeping the head pose unchanged, the system [8] synthesizes the novel view for the facial part then blending the new face area into the input image. The latter approach results in certain

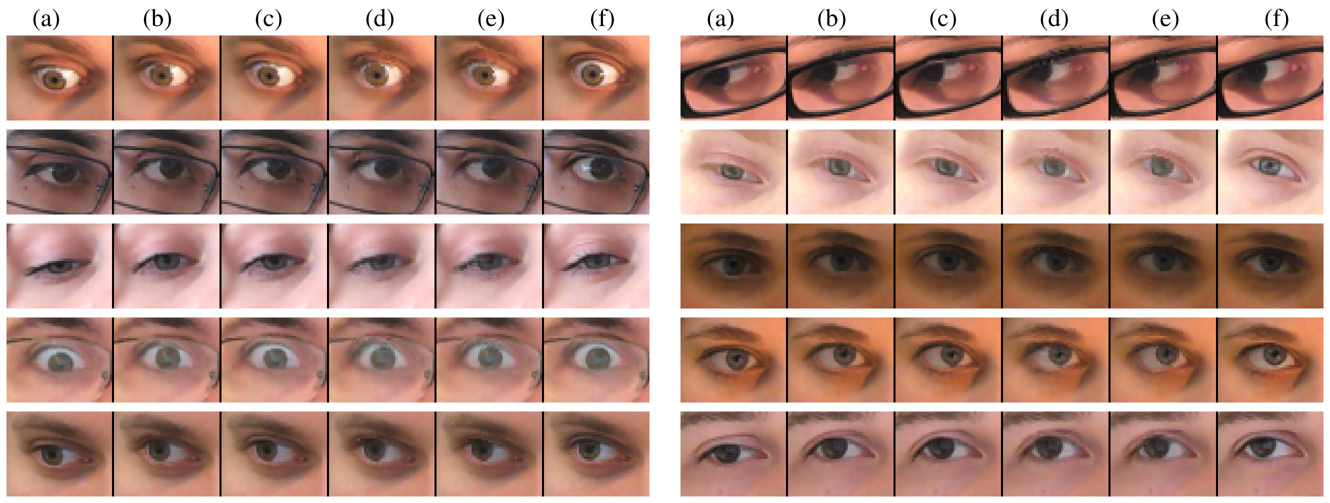


Fig. 12. Results on a random subset of the hold-out test set. From left to right: (a) Input, (b) Eye-flow forests, (c) Neural network supervised forests, (d) Coarse-to-fine warping with the lightness correction module, (e) Coarse-to-fine warping without the lightness correction module, (f) Ground truth. The full variant (CFW + LCM) of deep warp system (e) generally performs the best.

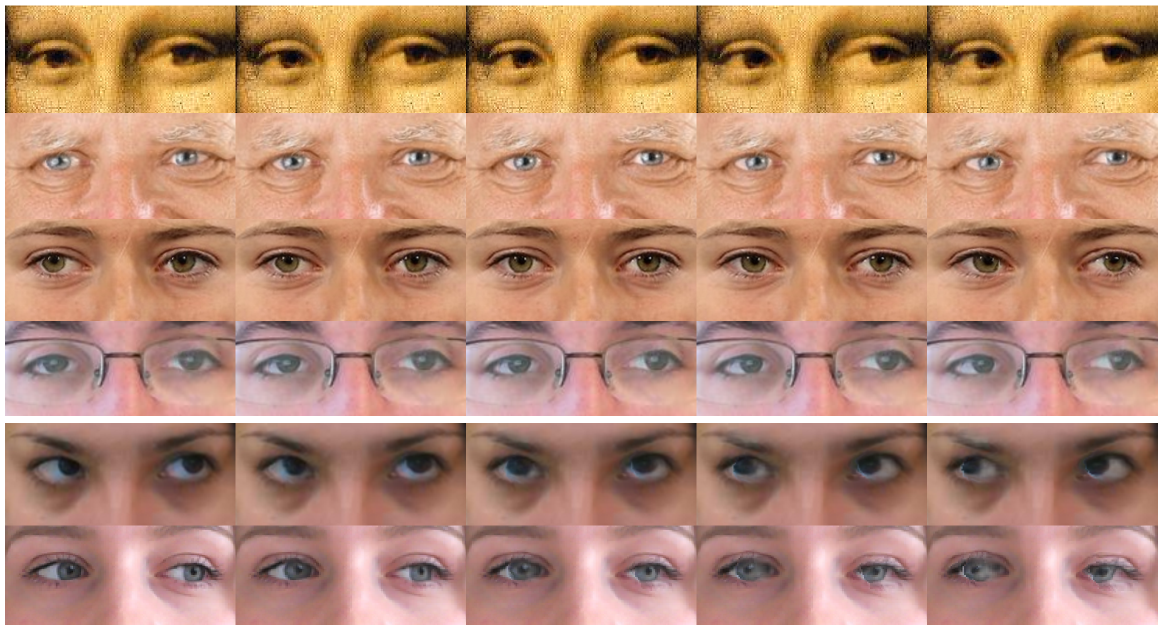


Fig. 13. **Horizontal redirection** with a model trained for both vertical and horizontal gaze redirection. For the first four rows the angle varies from $-15°$ to $15°$ relative to the central (input) image. The last two rows push the redirection to extreme angles (up to $45°$) breaking our model down.

distortion of face proportions. Moreover, [8] requires a simple per-person precalibration step and requires a (low-end) GPU for real-time operation (whereas our method achieves more than 30 fps on a single core of an Intel Core-i5 CPU).

Finally, the supplementary video [50] demonstrates a real-time screencast of our NNSRF system running on a variety of people under a variety of conditions typical for teleconferencing scenario. We also present animations, produced by our deep warp system trained simultaneously for vertical and horizontal redirection, in the **supplementary material**.

4.3 User study

To confirm the improvement corresponding to different aspects of the proposed models, which may not be adequately reflected by ℓ_2-measure, we performed a user study enrolling 41 subjects unrelated to computer vision to compare four methods (EFF, NNSF, CFW, CFW+LCM). Each user was shown 80 quadruplets of images, and in each quadruplet one of the images was obtained by re-synthesis with one of the methods, while the remaining three were unprocessed real face images. The example screen-shot from the user study interface is shown in Figure 16. In comparison to conference paper [44], we changed the setup to showing the full face image, while still instructing the users that only eye region will be resynthesized. Showing full face was motivated by more direct measurement of users' perception of the overall realism of the resulting images.

Twenty randomly sampled results from each of the compared methods were embedded in the set of quadruplets shown to each participant. The ordering of the methods and the position of the right answer were randomized. When a quadruplet was shown, the

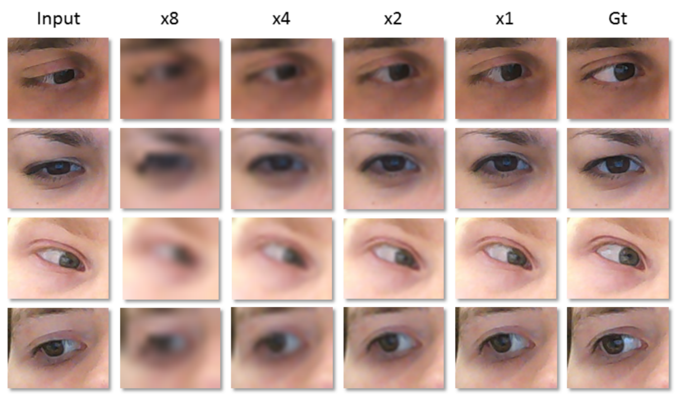


Fig. 14. The results of NNSF system for the $15°$ upwards redirection with input images of lower resolutions (the downsampling factors are shown at the top). Gaze redirection is persistent even for very low resolution images.

TABLE 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.

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

task of the subject was to click on the artificial (re-synthesized) image as quickly as possible. For each method, we then recorded the number of correct guesses out of 20 (for an ideal method the expected number would be 5, and for a very poor one it would be 20). We also recorded the time that the subject took to decide on each quadruplet (better method would take a longer time for spotting). Table 1 shows results of the experiment.

In general, all methods performed very well, approaching the best performance, which is 25% (the performance of random guess). The performances of the NN-supervised random forest and the full deep warp system are comparable, while the other two systems (eye flow forest and coarse-to-fine warping without lightness correction) performed a little worse. However, if considering only fast (confident) clicks, neural networks become slightly worse in comparison to eye flow forest, as well as to NN-supervised forest. Such performance of the deep architecture is explained by the fact, that it works on the fixed basic resolution, while the forest based methods could process images at the eye on native resolution of the cropped image. For a few examples, the users were apparently able to quickly notice the interpolation artifacts or the artifacts near the border in the deep warp system results. As a result, the teacher CFW method performs a bit worse than the student NNSF in these metrics. In terms of mean time that a user took for making a guess, deep warp architectures outperformed forest-based, because of the big contribution of the hardest samples, where users got stuck for a long time.

4.4 Computational speed and memory demands

Our 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 eye flow forest method requires from 3 to 30 ms to perform the remaining operations like querying the forest, picking optimal eye flow vectors, and performing replacements. The large variability is due to the fact that the bulk of operation is linear in the number of pixels we need to process, so the 30 ms figure corresponds to the situation with the face spanning the whole vertical dimension of the frame. Further trade-offs between the speed and the quality can be made if needed (e.g. reducing the number of trees twice will bring only very minor degradation in quality and almost two-fold speedup).

The neural network supervised forest method performs comparably to the eye 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 eye flow forest, because of the smaller tree depth, however this difference is not crucial in our implementations. The significant improvement is in memory consumption: eye flow forest method typically requires $100-200$ Mb, which should be stored in RAM memory at test-time, while the neural network supervised forest requires only $1.5 - 3$ Mb. The big difference in memory requirements occurs because of error distributions stored in leaves in the former case (Section 3.2) and only two-dimensional flow vector in the latter (Section 3.4).

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 our experiments), while also being universal, i.e. not tied to a specific redirection angle.

5 Summary and discussion

We have presented a new approach to gaze redirection based on supervised machine learning. The key advantages of the approach is its ability to work with a monocular input and in real-time on consumer-grade devices. The key novelty of our approach is the idea of using a large training set in order to learn how to redirect gaze. We have found a learnable and generalizable entity: the warping field of displacement vectors. It is used by our system within pixel-wise "copy-paste" operations.

We have presented three different systems that instantiate the approach. The two methods learn to predict the warping filed in a weakly supervised setting. The first variant of our system redirects

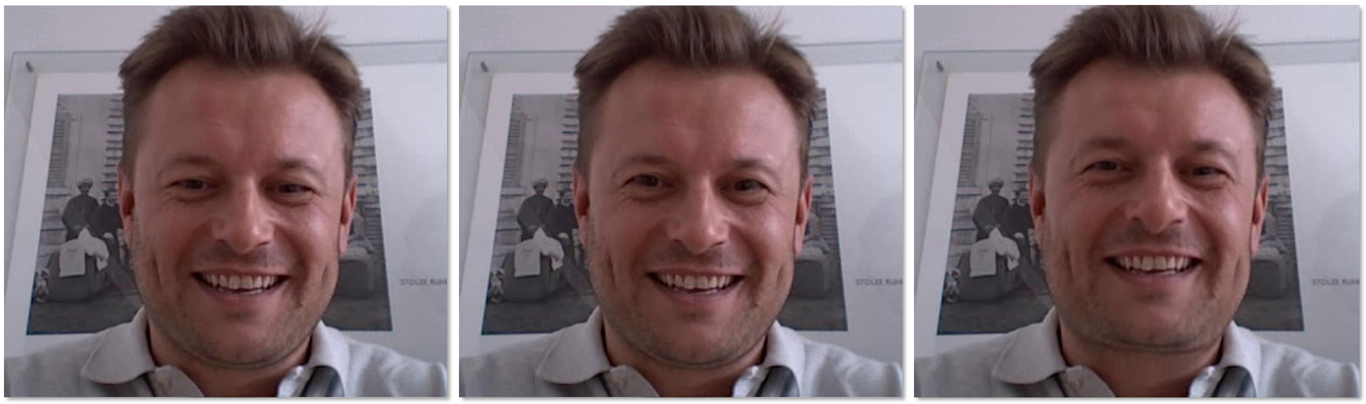


Fig. 15. Comparison with monocular gaze correction method from [8]. **Left** – the input video frame (taken from [8]). **Middle** – the output of our NNSF system. **Right** – the result of [8]. While both systems achieve convincing redirection effect, our NNSF system avoids the distortion of facial proportions (especially in the forehead and the chin regions), while also not requiring GPU to achieve real-time performance.

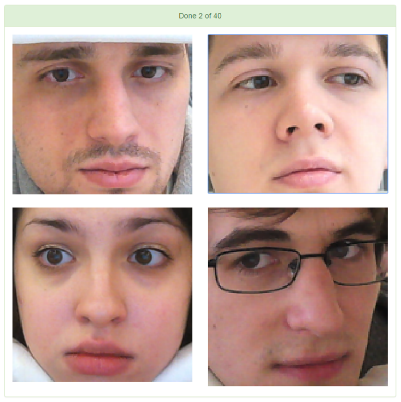


Fig. 16. The screen-shot from the user study interface. User was instructed, that one of the four images is not real, and was asked to click on the one, which seems unnatural, spending not much time (trying not to exceed 5 seconds). In this example, the top left image is the right answer.

the gaze by a fixed angle and runs in real-time on CPU (using a random forest predictor), while the second takes a redirection angle as an input and thus allows to change gaze continuously in a certain range, while also obtaining the higher quality result. This system predicts the warping field using a deep convolutional network with coarse-to-fine architecture of warping modules and embeds redirection angle and feature points as image-sized maps. In addition to warping, photorealism is increased using the lightness correction module. Finally, the third system can be regarded as a hybrid of the first two, as it essentially "condenses" the neural network into a random forest, achieving high-quality results, fast operation, and very compact models (all for a fixed redirection angle).

Notably, we have performed the user study showing the high photorealism of suggested methods, as the guess ratio is close to a random guess. Our user study of the system actually confirmed that unrealism in results is seldom caught by users, with very little artifacts being noticed. Arguably, our approach has successfully crossed the "uncanny valley" for the gaze redirection task.

Looking into the future, our approach gives greate promise of learning-based methods for photorealistic image synthesis and editing. An obvious next target is the extension of the proposed approach to face editing. For some face editing effects (e.g. wide smile), however, the warping-based model can fail to synthesize new content.

References

[1] L. J. Wallis, F. Range, C. A. Müller, S. Serisier, L. Huber, and Z. Virányi, "Training for eye contact modulates gaze following in dogs," *Animal behaviour*, vol. 106, pp. 27–35, 2015.

[2] C. L. Kleinke, "Gaze and eye contact: a research review." *Psychological bulletin*, vol. 100, no. 1, p. 78, 1986.

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

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

[5] T. Cham, S. Krishnamoorthy, and M. Jones, "Analogous view transfer for gaze correction in video sequences," in *Seventh International Conference on Control, Automation, Robotics and Vision, ICARCV 2002, Singapore, 2-5 December 2002, Proceedings*, 2002, pp. 1415–1420.

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

[7] C. Kuster, T. Popa, J.-C. Bazin, C. Gotsman, and M. Gross, "Gaze correction for home video conferencing," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 174:1–174:6, 2012.

[8] D. Giger, J.-C. Bazin, C. Kuster, T. Popa, and M. Gross, "Gaze correction with a single webcam," in *IEEE International Conference on Multimedia & Expo*, 2014.

[9] K.-I. Okada, F. Maeda, Y. Ichikawaa, and Y. Matsushita, "Multiparty videoconferencing at virtual social distance: Majic design," in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '94, 1994, pp. 385–393.

[10] A. Jones, M. Lang, G. Fyffe, X. Yu, J. Busch, I. McDowall, M. T. Bolas, and P. E. Debevec, "Achieving eye contact in a one-to-many 3D video teleconferencing system," *ACM Trans. Graph.*, vol. 28, no. 3, 2009.

[11] J. Zhu, R. Yang, and X. Xiang, "Eye contact in video conference via fusion of time-of-flight depth sensor and stereo," *3D Research*, vol. 2, no. 3, pp. 1–10, 2011.

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

[13] Z. Shu, E. Shechtman, D. Samaras, and S. Hadap, "Eyeopener: Editing eyes in the wild," *ACM Transactions on Graphics*, vol. 36, no. 1, Sep. 2016.

[14] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, pp. 1545–1588, 1997.

[15] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[16] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 775–781.

[17] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. J. V. Gool, "Random forests for real time 3d face analysis," *International Journal of Computer Vision*, vol. 101, no. 3, pp. 437–458, 2013.

[18] J. Shotton, R. B. Girshick, A. W. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, "Efficient human pose estimation from single depth images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2821–2840, 2013.

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

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

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

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

[23] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *CVPR*, 2015.

[24] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," in *CVPR*, 2015.

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

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

[27] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 262–270.

[28] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, pp. 1462–1471.

[29] T. Xue, J. Wu, K. Bouman, and B. Freeman, "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 91–99.

[30] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in Neural Information Processing Systems*, 2015, pp. 2539–2547.

[31] A. Ghodrati, X. Jia, M. Pedersoli, and T. Tuytelaars, "Towards automatic image editing: Learning to see another you," *CoRR*, vol. abs/1511.08446, 2015. [Online]. Available: http://arxiv.org/abs/1511.08446

[32] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Advances in neural information processing systems*, 2015, pp. 1252–1260.

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

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[35] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[36] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," in *European Conference on Computer Vision*. Springer, 2016, pp. 286–301.

[37] R. Yeh, Z. Liu, D. B. Goldman, and A. Agarwala, "Semantic facial expression editing using autoencoded flow," *CoRR*, vol. abs/1611.09961, 2016. [Online]. Available: http://arxiv.org/abs/1611.09961

[38] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," *CoRR*, vol. abs/1702.02463, 2017. [Online]. Available: http://arxiv.org/abs/1702.02463

[39] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg, "Transformation-grounded image generation network for novel 3d view synthesis," *CoRR*, vol. abs/1703.02921, 2017. [Online]. Available: http://arxiv.org/abs/1703.02921

[40] C. Bucilu, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.

[41] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2017. [Online]. Available: http://arxiv.org/abs/1503.02531

[42] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *CoRR*, vol. abs/1412.6550, 2014. [Online]. Available: http://arxiv.org/abs/1412.6550

[43] D. Kononenko and V. Lempitsky, "Learning to look up: realtime monocular gaze correction using machine learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4667–4675.

[44] Y. Ganin, D. Kononenko, D. Sungatullina, and V. Lempitsky, "Deepwarp: Photorealistic image resynthesis for gaze manipulation," in *European Conference on Computer Vision*. Springer, 2016, pp. 311–326.

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

[46] T. Baltrusaitis, P. Robinson, and L.-P. Morency, "Constrained local neural fields for robust facial landmark detection in the wild," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 354–361.

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

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[49] M. Oquab, "Torch7 modules for spatial transformer networks," https://github.com/qassemoquab/stnbhwd, 2015.

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

AUTHORS

Daniil Kononenko is a PhD candidate in the Computer Vision group at Skolkovo Institute of Science and Technology (Skoltech). Prior to that, he was a researcher at Datadvance. He completed BSc in 2011 and MSc in 2013 from Moscow Institute of Physics and Technology. His research interests are concerned with computer vision and machine learning.

Yaroslav Ganin is a PhD student in Montreal Institute for Learning Algorithms, Université de Montréal. He previously worked at the Computer Vision group at Skolkovo Institute of Science and Technology (Skoltech), and at NVIDIA. He completed MSc in 2011 from Lomonosov Moscow State University. His research interests are deep learning and computer vision.

Diana Sungatullina is a Junior Researcher in the Computer Vision group at Skolkovo Institute of Science and Technology (Skoltech). Prior to that, she was a Research Intern at Advanced Digital Sciences Center in Singapore. She completed MSc (Specialist) in 2014 from Lomonosov Moscow State University. Her research interests are concerned with computer vision, image processing, and deep learning.

Victor Lempitsky is an associate professor and the head of the Computer Vision group at Skolkovo Institute of Science and Technology (Skoltech), which is a new research university in Moscow area. Prior to that, he was a postdoctoral researcher at the Visual Geometry Group of Oxford University, and at the Computer Vision group of Microsoft Research Cambridge. He also was a researcher at Yandex, which is the biggest internet search company on the Russian market. Victor holds a PhD (”kandidat nauk”) from Moscow State University (2007). His research interests are concerned with computer vision, biomedical image analysis, and deep learning.