Project Report: Learning Polarization Difference Imaging

s6biulus
Department of Computer Science
University of Bonn
s6biulus@uni-bonn.de

Tim Oliver Sauermann
Department of Computer Science
University of Bonn
tim.sauermann@sauermannonline.de

Daniel Hubert Biskup
Department of Computer Science
University of Bonn
daniel@biskup.email

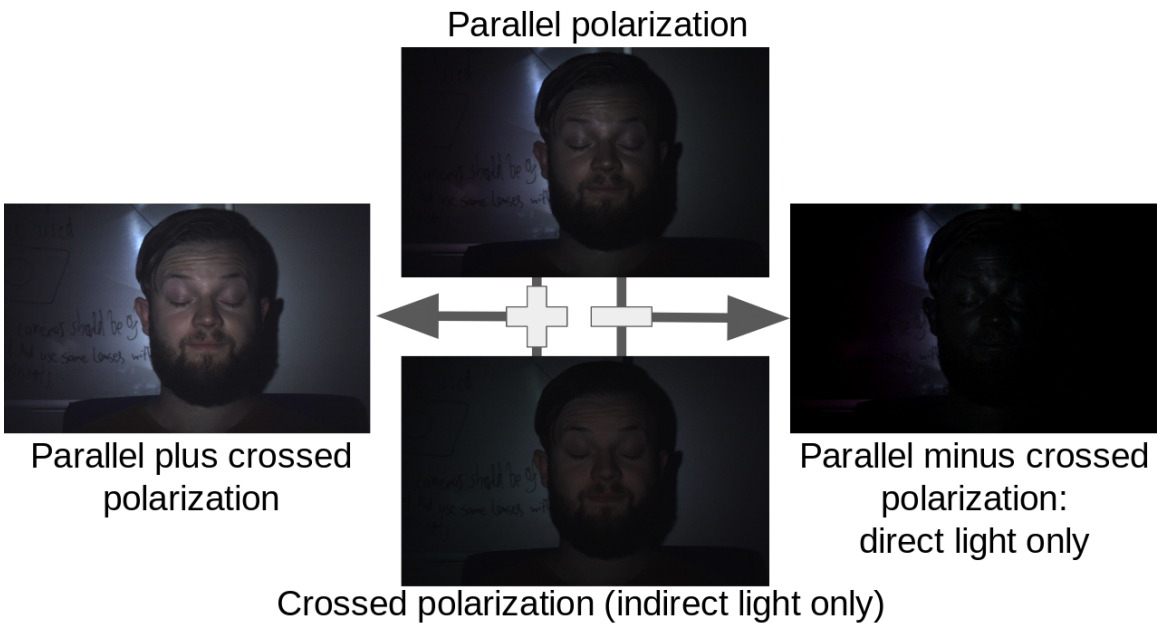

Figure 1: Polarization images captured by our setup (center), their sum (left) and the result of subtracting one form the other (right)

1 MOTIVATION

For some tasks it is required to get images that separate the direct and indirect reflected components of a light source illuminating a scene. This can be achieved by the technique of polarization difference imaging [Rowe et al. 1995]. Since this is a quiet time-consuming process that requires special hardware a more simple and effective way to get an image separated into this two components would be desirable. This project tries to realize that using deep learning mechanisms on normal pictures of human faces.

2 IMAGE ACQUISITION

Any project involving Deep Learning requires training data. To this end we devised a setup (see figure 5 and 6) which allows us to capture pairs of polarized images of human faces. An example of an image pair captured this way can be seen in figure 1.

Using duct tape, opaque black curtains were attached to the lab rooms windows as to ensure a consistently dark environment.

To capture a pair of polarization images, we take two images of the same static scene. For each pair, one image each is taken while the scene is illuminated with horizontally and vertically polarized light respectively. As a camera we use the *Flir Grasshopper 3* with a linear polarizing film in front of its lens (see figure 2).

Polarized light gets unpolarized when it's indirectly reflected from the subject, e.g. due to sub surface scattering in the human skin. In the case of direct reflection it keeps it's original polarization. Thus, when the filter in front of the camera absorbs all horizontally polarized light and we take an image of the scene under horizontally polarized illumination (i.e. the polarization of the light source and the camera are parallel to each other) the camera will receive the full amount of directly reflected light but just the parallel components of the unpolarized light (see figure 1 top). On the other hand, when the scene is illuminated with with vertically polarized light (i.e. light and filter are cross polarized to each other) no directly reflected light (i.e vertically polarized light) will pass the horizontal polarization filter in front of the camera, while the unpolarized (i.e. not directly reflected light) will (see figure 1 bottom).

This fact can be exploited by subtracting the cross polarized image from the parallel polarized image which yields an image that contains only the directly reflected light (see figure 1 right). Adding the cross polarized image and the parallel polarized image up yields an naturally looking image (see figure 1 left).

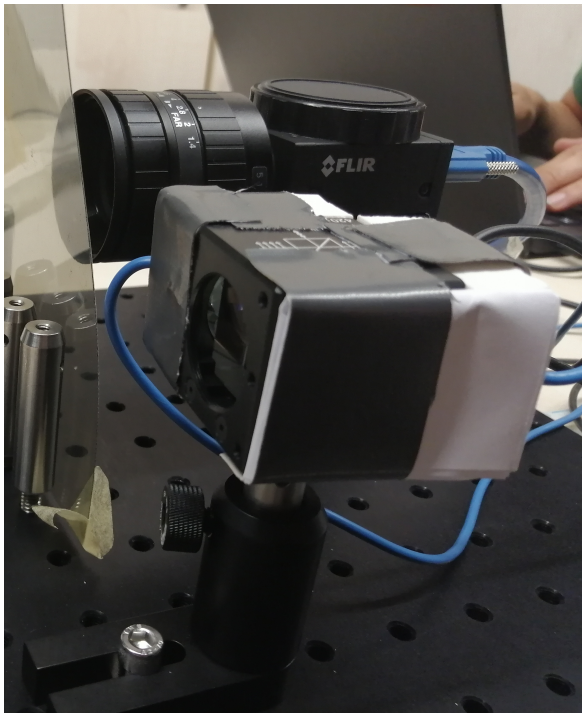

Figure 2: Camera with polarizing filter and LEDs attatched to polarizing beamsplitter cube

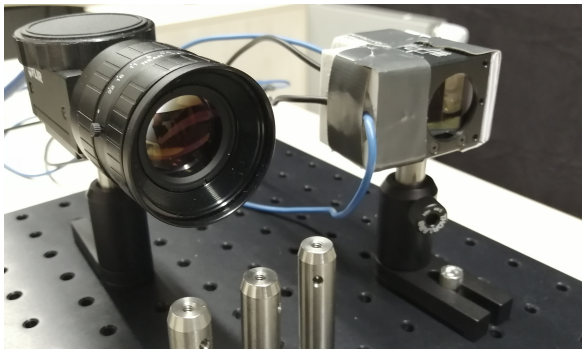

Figure 3: Camera and light source

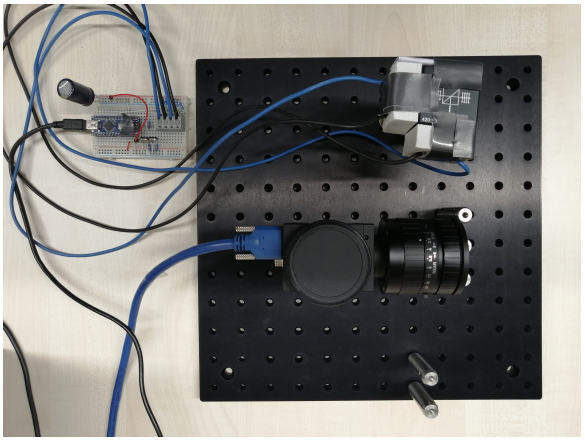

Figure 4: Setup from above. The Arduino to control the LEDs can be seen on the left.

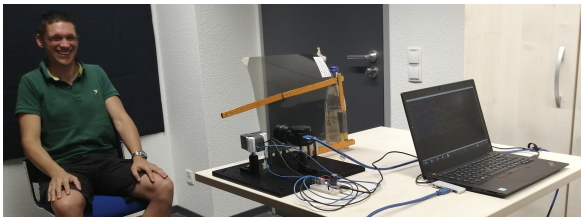

Figure 5: Image acquisition setup

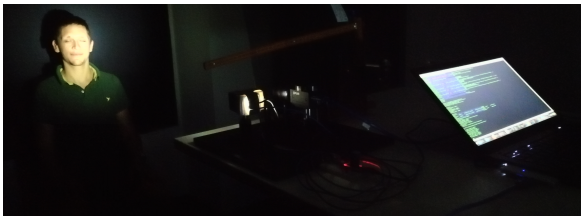

Figure 6: Image acquisition setup (in use)

To be able to alternate between horizontally and vertically polarized illumination of the scene, we use two identical LED-chips of type *Samsung LH351B-RT*, attached to a star-board for cooling and connect them to a computer via an *Arduino Nano* which is controlling a simple mosfet-driver and in that way enables us to control them using a Python script. We favoured a simple driver like this over a dedicated constant-current driver to enable us to use high switching speeds. The LEDs are mounted on lenses collimating the beams to an angle of 8° and attached to a polarizing beamsplitter cube of type *ThorLabs CCM1-PBS251* using custom 3D printed mounting brackets and duct tape (see figure 3).

To ensure, that two images of the same pair are aligned to each other as good as possible the subject is not allowed to move (e.g. change it's facial expression) in between the two shots belonging to the same pair of polarization images. To minimize the risk the two images are taken rapidly after another using the internal buffer of the camera. To still allow for a reasonably fast acquisition images of different facial expressions or directions we give the subject about one second of time to move between taking image pairs, this time is also used to transfer the pictures from the camera to the PC and save them to disk. This allows for an effective acquisition rate of about 3600 polarization image pairs an hour.

An alternative way to using two polarized LEDs and a single camera with a polarization filter in front of it's lens would have been to use one polarized LED and two cameras, both attached to the polarizing-beamsplitter cube. This would have allowed us to capture a huge data set faster, since both cameras can be triggered simultaneously (by having one software triggered camera

hardware triggering the other) which remedies the problem of misalignment between two images of the same pair due to movement of the captured subject. Then the subject could just slowly move its head while the cameras produce multiple images per second. In contrast, the image acquisition rate of our setup is much slower. Even though this method of acquisition is potentially faster then ours we decided against it since using two cameras comes with its own set of technically intricate challenges. Most noteworthy, the two cameras have to be perfectly aligned to each other. This is very hard to archive and worthy of a project on its own. Furthermore a continually moving object would introduce motion blur into the images, potentially requiring the use of de-blurring techniques which would make the pre-computations even more complex. This problem gets even more noticeable since the power of the required point-light-source can't exceed a certain limit without the risk of taking severe damage to the subjects eyes.

Following is a list of the technical equipment used:

- Camera: *Flir Grasshopper 3, model: GS3-U3-23S6C-C*
- Two LEDs: *Samsung LH351B-RT Chips 5000K @ 1200mA 3,1V 3,8W*
- LED controller: *Arduino Nano*
- Broadband Polarizing Beamsplitter Cube: *ThorLabs CCM1-PBS251/M*
- Linear polarizing film: *Screen-tech ST-38-20 - Linear Polarisator*
- Duct Tape: *Ferociously Strong T-Rex Duct Tape*

3 LEARNING

The goal of the learning part of this project is to train a network which outputs either a parallel polarized image or a crossed polarized image from a given input image. This problem is very similar to Image-Denoising in a sense that one of the polarized images could be thought of as noise and that way could be removed from the input image, so that the other polarized image remains. This way both polarized images are accessible. In order to solve this problem, we combined ResNet- and Auto-Encoder-architectures.

We used an auto-encoder-architecture, which is a neural network architecture, that combines convolution and deconvolution layers. The convolution layers with 2 strides downsample the image in every layer, whereas the deconvolution layers with 2 strides upsample the image. While the convolution layers act like feature extractors, the deconvolution layers recover the details of the image content. Unlike segmentation problems, we did not use Max Pooling between layers in order to downsample, since recovering minor details of the image is very important in the given context. Generally deeper networks produce better results, since the network can express more complex non-linearities as the layers increase. In conclusion we wanted to use a very deep network for solving this problem, which is opposed by the problem that deep networks suffer from the problem of vanishing gradients. Skip connections, or residuals, in a ResNet architecture can solve this problem to some extend. The skip connections in ResNet feed the activation unit of a layer directly to a deeper layer of the network. With the addition of the skip connections connecting every two convolutional layers to their corresponding, mirrored deconvolutional layers, our network will be less affected by the vanishing gradient problem (see Figure 7).

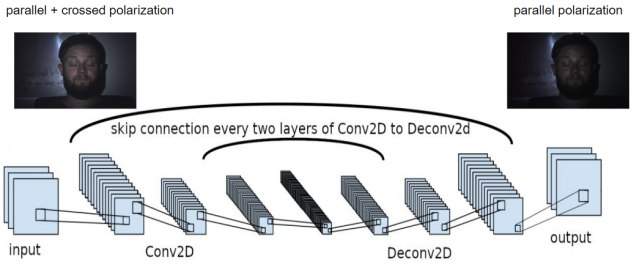

Figure 7: The architecture of the network

Since the data collection process takes a long time for our problem, we created a mock-dataset to create an initial network and train it. The idea is to create a mock-dataset which consists of compositions of an orange image with black background and an cat image, so that these could be used to train the initial network (see Figure 8). Since the operation to separate this overlaid images is equivalent to the operation we are planning to do with our actual problem, then we can use the same network for our original problem after verifying that it is possible to train the network successfully. Only slight by changes of its parameters would be necessary. The mock-dataset consists of 12,500 cat-images, composed with orange-images. The cat images were obtained from the "Dogs vs. Cats"-dataset from https://kaggle.com.

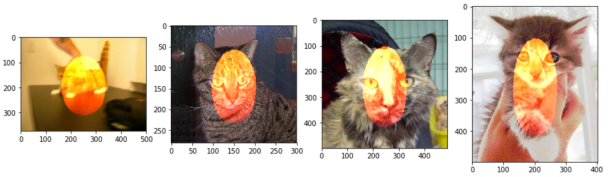

Figure 8: The mock dataset

ACKNOWLEDGMENTS

We thank Prof. Dr. Matthias B. Hullin for giving us the opportunity to work on this project. Furthermore we want to thank *AG Hullin* and *AG KLEIN* for providing required technical equipment as well as a room to set up and conduct our experiments in.

The code we wrote for triggering the cameras, controlling the LEDs and training the neural networ is provided in a repository at https://github.com/DanielBiskup/ComputationalPhotographyProject.

REFERENCES

Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley. 2017. Removing rain from single images via a deep detail network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3855–3863.

Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. 2016a. Image Denoising Using Very Deep Fully Convolutional Encoder-Decoder Networks with Symmetric Skip Connections. *CoRR* abs/1603.09056 (2016). arXiv:1603.09056 http://arxiv.org/abs/1603.09056

Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. 2016b. Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections. *CoRR* abs/1606.08921 (2016). arXiv:1606.08921 http://arxiv.org/abs/1606.08921

Marco Peixeiro. 2019. Hitchhiker's Guide to Residual Networks (ResNet) in Keras. (April 2019). Retrieved July 12, 2019 from https://towardsdatascience.com/hitchhikers-guide-to-residual-networks-resnet-in-keras-385ec01ec8ff

M. P. Rowe, E. N. Pugh, J. S. Tyo, and N. Engheta. 1995. Polarization-difference imaging: a biologically inspired technique for observation through scattering media. *Opt. Lett.* 20, 6 (Mar 1995), 608–610. https://doi.org/10.1364/OL.20.000608