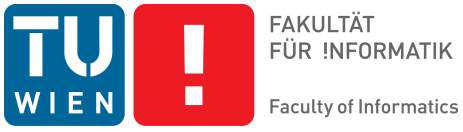


101.388 Wahlpflicht-Projekt

Shoeprint Matching with multichannel normalized cross correlation

Faustmann Erik

January 18, 2021

Computer Vision Lab

Institute of Visual Computing & Human-Centered Technology TU Wien

Autonomous Systems Center for Vision,

Automation & Control AIT - Austrian Institute of Technology

Supervisor: Manuel Keglevic

Contents

1 Abstract

Footwear impressions can often be found at crime scenes, but the quality and variety of these crime scene impressions makes their manual analysis a laborious task for the forensic footwear examiner. A big issue is, that the patterns of these crime scene impressions can be hard to distinguish from the background. The goal of this research was therefore the implementation of a matching algorithm for crime scene impressions and reference images from a database. The algorithm calculates multiple feature maps generated by a pre-trained neural network and then uses normalized cross correlation to compute a correlation matrix. The correlation matrix consists of rows representing the crime scene impressions and the columns represent the reference images. The highest correlation coefficient for each row represents the final match according to the algorithm. This script shall eventually assist the examiner. Furthermore we put some effort into finding ways to increase the performance of this algorithm.
For the algorithm we used a pre-trained neural network (the first three convolution layers of googlenet) for feature generation in combination with a function, that calculates the normalized cross correlation between the features of the crime scene impressions and the reference features from the database. To achieve better performance, we cropped out parts of the features that don't include valuable information, but lead to artifacts in the correlation map. We further investigate the effects of rotating either the reference image or the crime scene impression image in connection with the correlation scores, using the best score from a chosen interval of rotation angles. In order to reduce the calculation time we only used a subset of the online available FID-300 dataset, consisting of 50 crime scene impression images and their corresponding reference images.
The cropping of features does not show a significant improvement, but in combination with rotating the reference images it leads to better matching results.
Conclusion: We were able to run a functioning implementation of a shoeprint matching algorithm in pytorch using multiple feature channels and normalized cross correlation. Our results showed satisfying performance with room for further improvement.

2 Introduction

Footwear impressions play a significant role as evidence in the investigation of crime scenes. Forensic experts have to assign aquired crime scene impressions to reference impressions in order to retrieve more information. Those crime scene impressions are mostly provided by using so called gel lifters (see Figure 1). Gel lifters have a sticky porous surface. The gel lifter is pressed onto the surface that is holding the impression. The porous gelatin surface penetrates into the pores of the impression-held surface and the dust particles stick to it. When removing the lifter from the surface the dust remains on the lifter, because of its adhesive characteristics. After the removal lifted marks are then scanned or photographed. A problem with gel lifter is, that they also pick up surrounding contaminating material, such as dust or other particles, which can cause making parts of the impression unrecognizable. In some cases the crime scene impressions are photographed on the spot, which makes matching in the later stage difficult. The same methods can be used for aquiring the reference impressions.

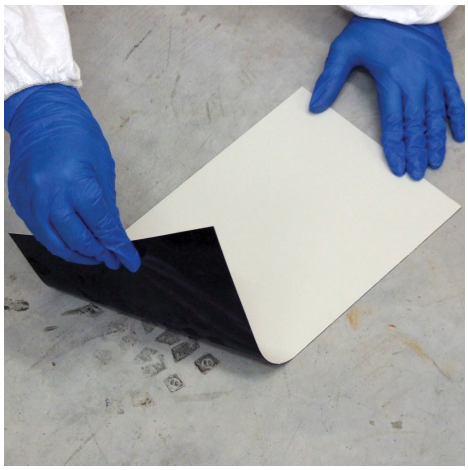


Figure 1: Application of a Gel Lifter

For impressions in soft surfaces like soil, snow or sand, casting is the most common method for collecting crime scene impressions. Casting uses a powdered stone material, for example dental stone mixed with water. This mixture is then poured into the impression. Imprints can be further processed to bring out more details. Programs such as Adobe Photoshop can be used to improve the quality. Another option is using chemical stains and or dyes to enhance image color or increase the the contrast against the background[1]. This method is shown in Figure 2.

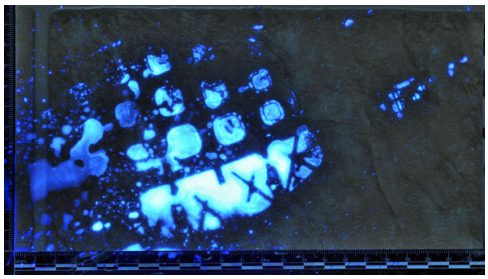


Figure 2: A faint blood shoe impression on lineoleum, enhanced by treatmeant with a chemical

[1]http://www.forensicsciencesimplified.org/

When examining the footwear impression, we differentiate between class- and identifying characteristics[2]. Class characteristics are properties such as physical size, design and mold characteristics. Identifying characteristics on the other hand do not result from the manufacturing process, but come from different types of wear while using the shoes, for example marks caused by cuts, nicks, gouges and scratches. First, the examiner investigates the class characteristics between the crime scene print and the known shoe. If no inconsistencies occur during this process, the examiner then looks for identifying characteristics in both shoe images. The fact that occurrence, size, shape and postition are unpredictable, leads to a low probability of recurrence in the same manner on a different shoe. Thus, one single identifying characteristic combined with the class characteristics is a strong indication for the identification of a match.

Additionally, a manufacturer may use different molds to produce the same outsole design for the same model and the same size of the shoe. In Figure 3, three Nike Air Force I oustoles with the same general outsole design features are shown, but some areas of the sole show slight variations in the mold between these three shoes (see Figure 4). These varations include design differences, stippling or other texturing and positioning of the logo [KSRF]. Identifying a particular mold, reduces the potential population shoes of this design and size.

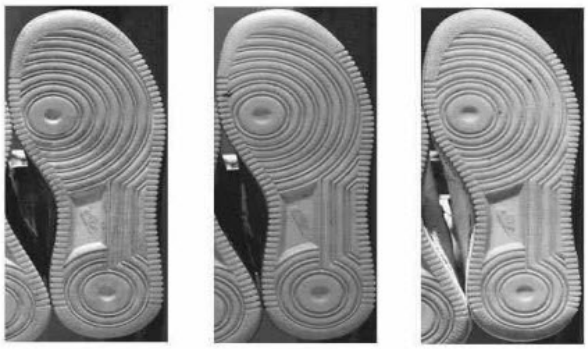


Figure 3: Three Nike Air Force I outsoles with the same general outsole design features.

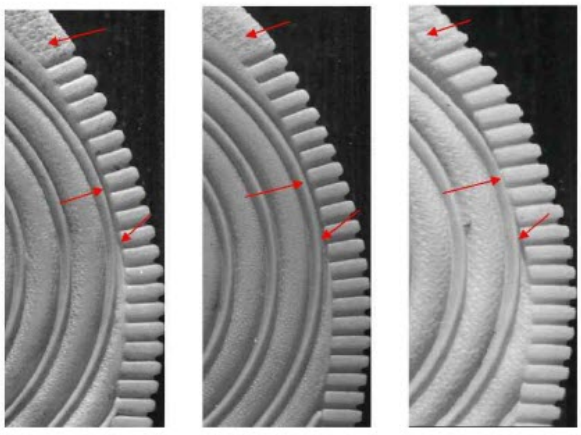


Figure 4: Enlarged areas of the outsoles from Figure 3. Three arrows on each outsole depict the mold variations in each each shoe.

[2]https://archives.fbi.gov

Searching a questioned crime scene impression through a database with thousands of reference images is a tedious task and thus an automated system for matching crime scene prints to reference images is strongly desired. At this point in time, there is no automated system that can assist in this part of investigation. One challenge of this task, for example, is the differentiation between the actual impression of the shoe and the background, that comes along with the gel lifting process. A few of these examples are shown in Figure 5.

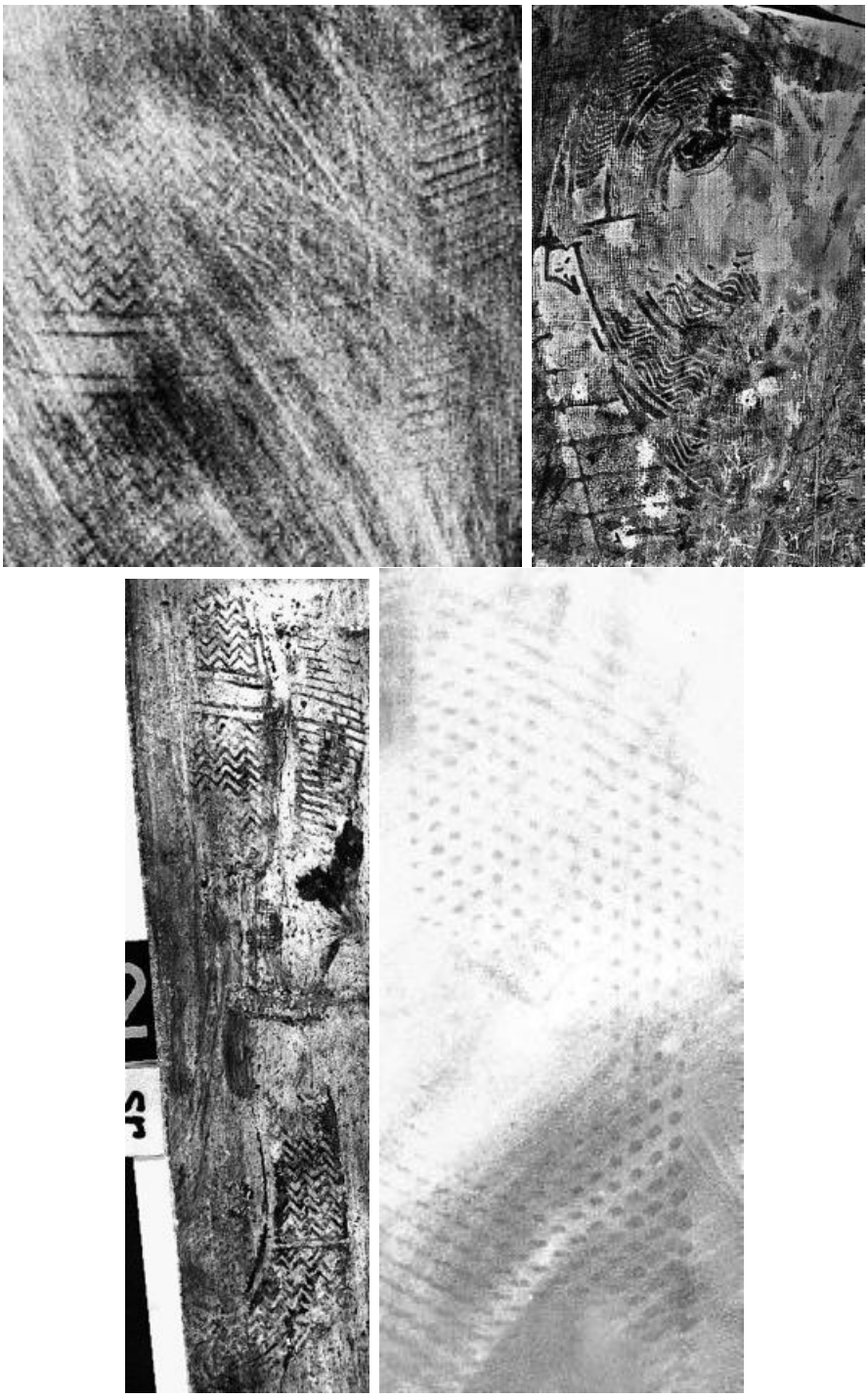


Figure 5: Crime scene impressions

Many different papers haven been published, introducing different ideas for automating the process of matching shoe prints. One paper by Kong, published in 2019 introduced the use of normalized cross correlation (NCC) on multiple feature channels, embedded in a siamese network. This model showed better results than any of the current state of the art methods. The model has been implemented in Matlab. For our research, we took the idea of using multi-channel normalized cross-correlation (MCNCC) and implemented the model in pytorch. We also present different ideas to increase the performance of matching crime scene impressions to clear reference impressions. Both sets of images are provided by an open accessable database named FID-300[3]. Figure 6 shows a crime scene impression, which is searched for in the database, that contains the reference images. This database also includes a label table, meaning that every track has a corresponding reference image.

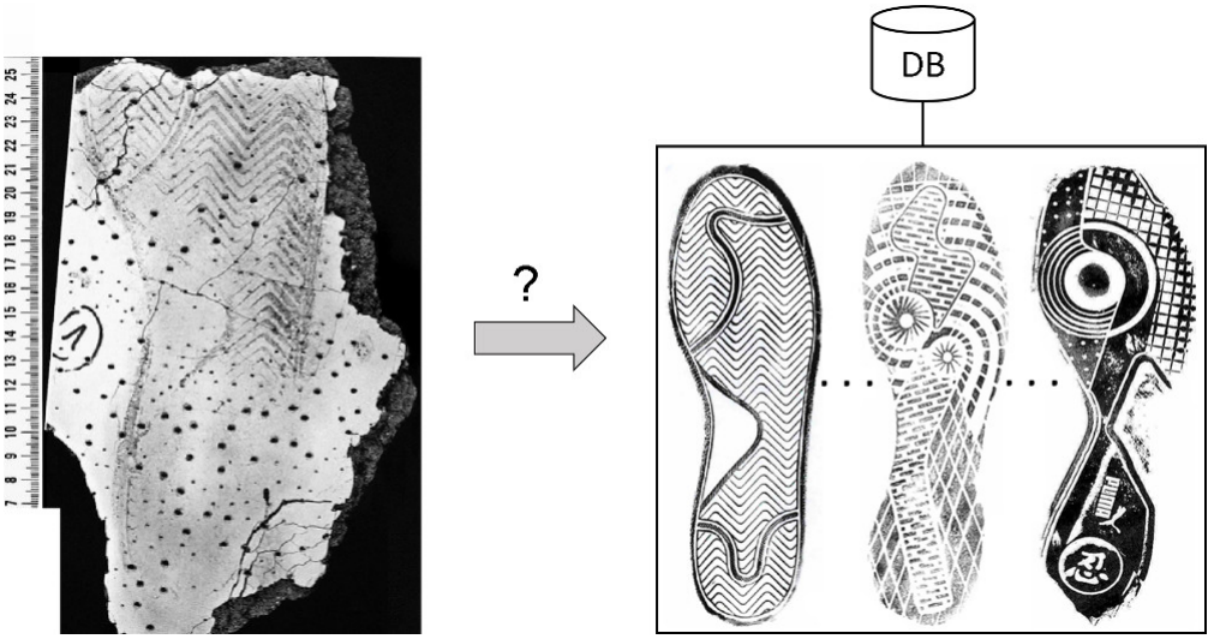


Figure 6: Crime scene prints and reference images

These kind of networks have been proven to work very well for all kinds of different image classification tasks. For this work, we study the differences between different types of normalization and their results. Furthermore, we investigate the effect of pre-processing methods to increase the performance of matching and also try different approaches of rotating the images to see the effect on the resulting correlation scores. Rotating all the images increases the computational time by a multiplicative factor and therefore makes this tasks additionally time consuming. In order to simplify the process, we used a smaller subset of the FID-300 database, containig only 50 crime scene impressions and the corresponding reference images. The recieved results on a subset of referenc images show some improvement, but are unfortunately still far from the expected outcome.

[3]https://fid.dmi.unibas.ch/

3 Related Work

3.1 Probabilistic Compositional Active Basis Models for Robust Pattern Recognition

There have been various approaches for shoeprint-matching or any similar type of matching between two differen domains. One of the most prominent works in the field of shoe print matching in recent years, has been done by Adam Kortylewski and Thomas Vetter in 2016 [KV]. In this paper, the robustness of hierarchical compositonal models (HCM) under challenging conditions is studied. First, a greedy EM-type algorithm to automatically infer the structure of compositional active basis models (CABMs) is introduced. Then the proposed representation and its learning process in a fully probabilistic manner is formulated. Finally the CABM is augmented with an implicit geometric background model, that reduces the models sensitivity to pattern occlusions and background clutter (see Figure 7).

(a) (b) (c) (d)

Figure 7: a) A non-linearly distorted probe image; b) the corresponding reference impression; c) a sketch of the learned CABM for the reference impression; d) An overlay of the CABM over the probe image [S]

The evaluation is done on a complex forensic image analysis task. The experimental results showed that the image analysis task is processed with an exceptional quality (Figure 8). We will later compare these results from Kortylewski with our own.

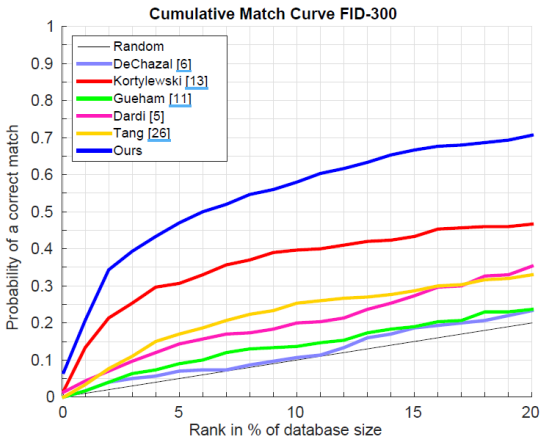


Figure 8: Results on the FID-300 dataset

3.2 Cross-Domain Image Matching with Deep Feature Maps

This paper proposes the use of normalized cross-correlation [KSRF] with multiple feature channels from a pre-trained convolutional neural network embedded in a siamese neural network and analyzes its effectiveness (see Figure 9). This method shows improved results compared to other state of the art approaches, like Kortylewski's method which was discussed previously.

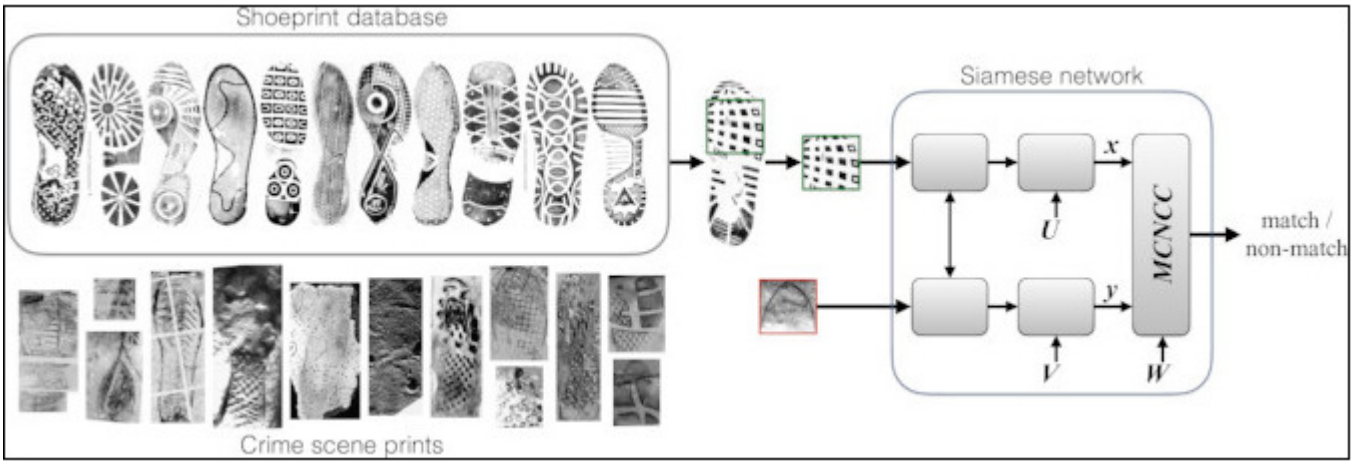


Figure 9: Siamese network for shoeprint matching; U and V are linear projection parameters for laboratory test impressions and crime scene photo domain respectively. [S]

The following formulas show the mathematical principles of Kong's approach.

Normalized Cross Correlation

$$NCC = \frac{1}{\mid P \mid} \sum_{i \epsilon P} \frac{(x\,[i] - \mu_x)}{\sqrt{\sigma_{xx}}} \frac{(y\,[i] - \mu_y)}{\sqrt{\sigma_{yy}}} \tag{1}$$

The pixels x from an image patch X are corrupted by noise and the pixels y are from another patch Y. P refers to the set of pixel positions in a patch.

Mulitchannel Cross Correlation

$$MCNCC = \frac{1}{N \mid P \mid} \sum_{c=1}^{N} \sum_{i \epsilon P} \frac{(x\,[i] - \mu_x)}{\sqrt{\sigma_{xx}}} \frac{(y\,[i] - \mu_y)}{\sqrt{\sigma_{yy}}} \tag{2}$$

With N as the number of feature channels, for example N = 3 for a RGB image.
MCNCC in combination with CCA provides an effective building block for a simaese network model. The experiments with this network have shown that even with very limited data, robust cross domain matching is achievable (Figure 10). Therefor we chose this work as starting point for our research paper.

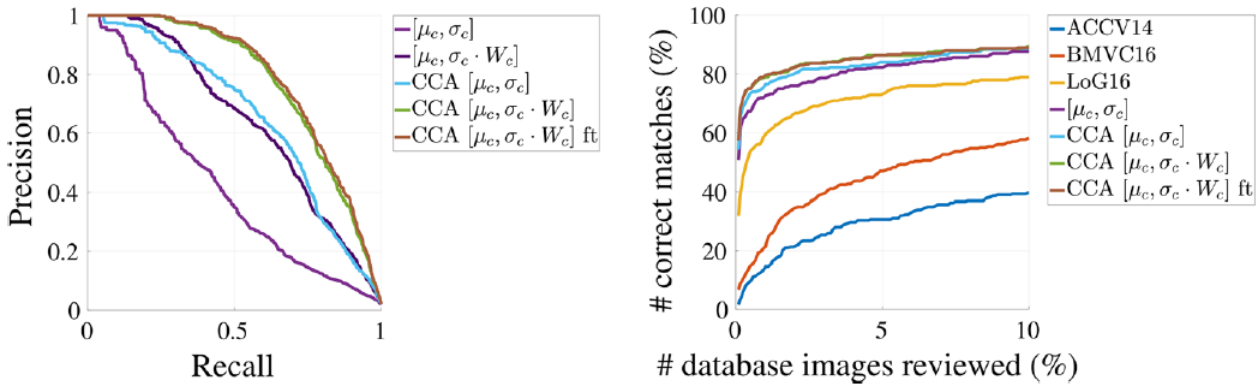


Figure 10: Comparing MCNCC with uniform weights ([μc, σc]), learned per-channel weights [μc, σc · Wc], learned linear projections (CCA [μc, σc]), piece-wise learned projection and per-channel weights (CCA [μc, σc · Wc]), and jointly learned projection and per-channel weights (CCA [μc, σc · Wc]

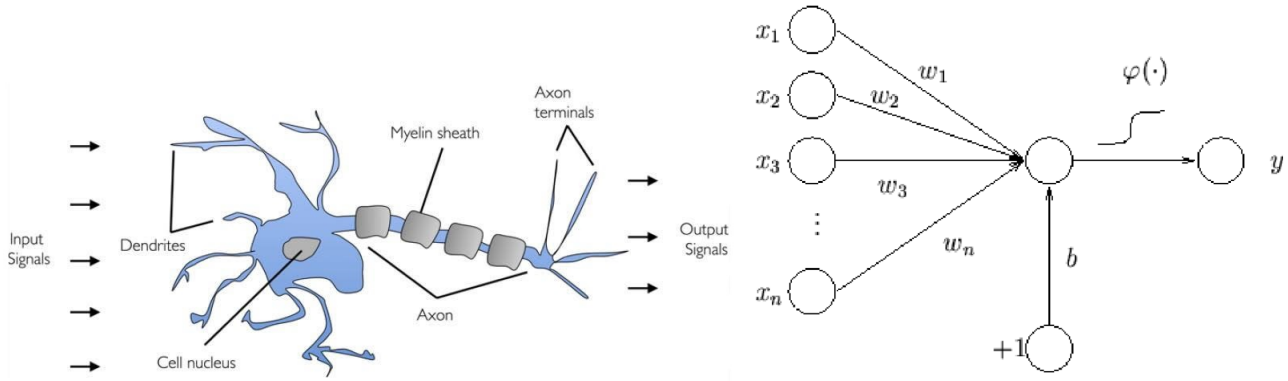
4 Methodolody

4.1 Basics of Deep Learning

Deep Learning is one of many machine learning methods and it is based on artificial neural networks. There are a number of different architectures such as deep neural networks, deep believe networks (DBN), recurrent networks (RN) and convolutional neural networks (CNN). It can be applied to various fields inlcuding computer vision, speech recognition, bioinformatics and many others with extraordinary results. Its origin dates back to the early 50's, but had its resurgance due to the availability of big data, better hardware (GPU's) and software (new models and toolboxes).

4.1.1 Perceptron

The perceptron is an algorithm, first introduced in 1958 by Frank Rosenblatt, for supervised learning of binary classifiers. It is a simplfied model of a biological neuron (Figure 11). Supervised learning means, that the classifier-function has to be trained by example input and ouput pairs.



"The embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence." - The New York Times, 1958

Figure 11: Left: Biological Neuron; Middle: Perceptron; Right: Binary Classification Task

$$y = \varphi(w_0 + \sum x_i w_i) \tag{3}$$

$$y = \varphi(w_0 + X^T W) \tag{4}$$

In Figure 11, we see the general architecture of a perceptron. The input vector **X** is connected with the weight vector **W** of the same dimension via the dot product. Also a bias valuew_0 is added to the term. The output comes from putting the resulting term into an activation function, for example a sigmoid function. That means that the resulting value y lies between 0 and 1, depending on the sign of the argument, hence why the perceptron is a binary classifier. The weights can be arbitrary initalized. Their purpose is to adjust during the so called training phase, so that the algorithm gets more and more accurate after each step of the training. During the training phase we feed the perceptron with inputs from a given dataset, while also providing the output values. This means we can calculate an error between the predicted outcome of the perceptron and the actual output from the dataset. Depending on the data we can use different Lossfunctions such as Empirical Loss, Binary Cross Entropy Loss, Mean Squared Error Loss (3) and so on.

$$J(W) = \frac{1}{n}\sum(y_i - \varphi(w_0 + X^T W)) \tag{5}$$

The goal is now to find network weights that achieve the lowest loss. The process of finding the best weights is called loss optimization.

$$W^* = argmin\frac{1}{n}\sum \mathcal{L}\varphi(x_i W, y_i) \tag{6}$$

4.1.2 Deep Neural Network

Linking mulitple perceptrons together as seen in Figure 12, leads to layer based structure divided into several layers. The first layer is called the input layer, the last layer is called the output layer and the layers inbetween are called hidden layers. Every layer consists of a number of perceptrons. If all perceptrons in a layer are each connected with all perceptrons from the previous layer, it is called a dense layer.

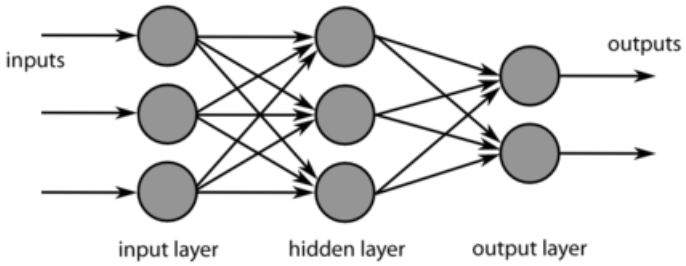


Figure 12: Neural Network with one hidden layer and two output neurons

Training

For the training step the weights or parameters of the neurons have to be adjusted, so that the loss reaches a minimum. In the beginning, the weights can be randomly initialized. After the first training step, the gradient of the loss function is calculated (formula 7). Then the weights get updated according to formula 8, where η describes the learning rate. Applying this for all the weights via the chain rule is called backpropation.

$$\frac{\partial J(W)}{\partial W} \tag{7}$$

$$W \leftarrow W - \eta\frac{\partial J(W)}{\partial W} \tag{8}$$

Choosing the right learning rate depends on the shape of the lossfunction. A small learning rate can converge and get stuck at a false minima and a large learning rate can overshoot and become unstable and diverge. So for complex networks it can be useful to use a learning rate that adapts to the landscape. There are many different gradient descent algorithms like SGD, Adam, Adadelta, Adagrad and RMSProp to name a few.

It can be beneficial to train in so called Mini-batches. The advantages are parallel computation and therefore a significant speed increase with the GPU. It can also lead to a more accurate estimation of the gradient and allow for larger learning rates. One problem that can occur while

training is overfitting. This means that the model is getting too complex and does not generalize well. Regularization is a method to workaround that problem. Two main methods of regularization are drop-out and early stopping. When using dropout the network randomly sets some activations to zero. The most common probability is 50%, so half of the neurons in the layer get deactivated. This forces the network not to rely on any single node. The second regularization method is stopping the training before there is any chance to overfit. In order to achieve this, it is necessary to inspect both training loss and test loss for every epoche during the training phase. The training is stopped when the loss on the test set is starting to grow again.

4.1.3 Convolutional Neural Networks

Convolutional neural networks are a class of deep learnin networks which came to prominence in 2012 after Alex Krizhevsky won that years ImageNet competition, dropping the error rate from 26% to 15%. The most popular use case for these networks is image classification or matching. For the classification task an input image is given to the network and as a result the model outputs a probabilty of pre-defined classes. The input image arrives the network as an array of pixel values. In most cases it is a three-dimensional array for RGB-images, where the first two dimensions are the height and width of the image and the third dimension refers to the color channels. Next, this array passes through a number of convolutional layers. Each layer has a certain number of filters, which itself are again arrays with certain values in it. These values are called weights or parameters and they can be initialized randomly. The depth of the filter has to be of the same depth as the input array. As described in Figure 13, the filter slides step by step through the input array beginning on the top left . Within each step an element-wise multiplication is being done. These mulitplications are then summed up and the result is a single number in the output array. This output array is called a feature map. Without any padding and an input array with the size of $n\,x\,n$ and a filter size of $m\,x\,m$, this leads to an output array with the size of $(n-m)\,x\,(n-m)$.

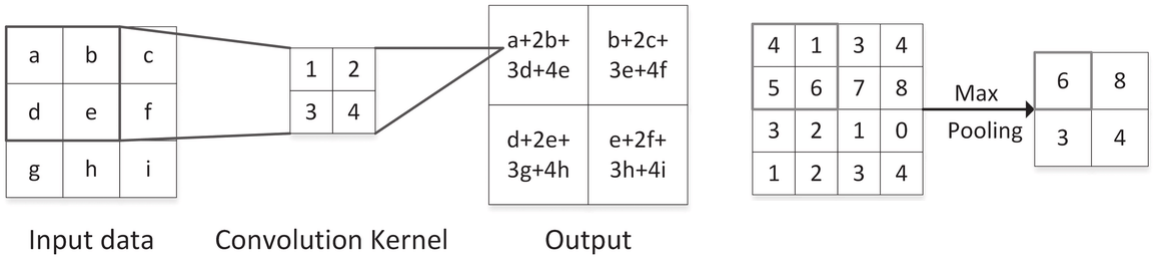


Figure 13: Convolutional filters and max pooling [S]

The task of these filters is to detect certain features. Features can be straight edges (horizontal, vertical, ect.), colors and curves for example (see Figure 14).

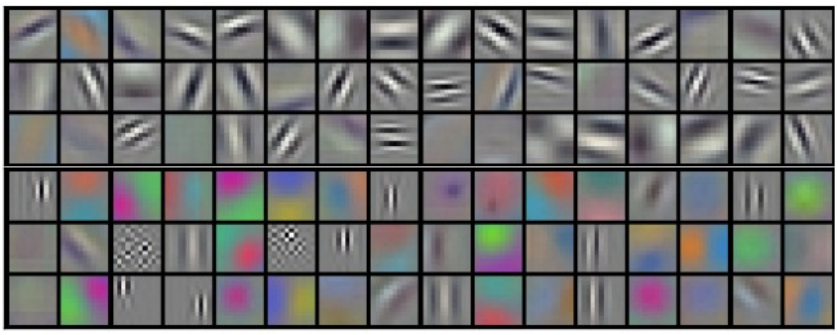


Figure 14: Weights

With each convolutional layer the activation maps represent more and more complex features. At the end of the network, a fully connected layer is attached. This layer takes the input volume and outputs a N-dimensional vector, where N is the number of classes. For a digit classification, N would be 10 for each digit. The numbers within the N-dimensionl vector represent the probabilty of a certain class.

4.1.4 Transfer Learning

Transfer learning in machine learning, focuses on applying knowledge gained from solving one problem to a different, but related problem. For our model we used the first three convolution layers from the GoogLeNet (also called Inception v1) [L], which won the ImageNet Large-Scale Visual Recognition Challenge 2014 (Figure 15).

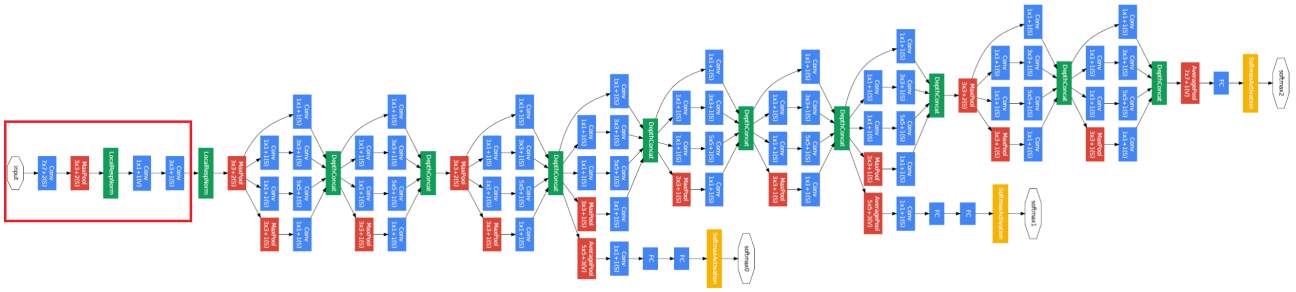


Figure 15: GoogLeNet Network (From Left to Right)

After the 3x3 Convolution layer (Figure 16), we get 192 feature maps, on which we perform normalized cross-correlation. The resulting correlation score matrices are then added together and divided by the number of channels (equation 2, chaper 3.2).

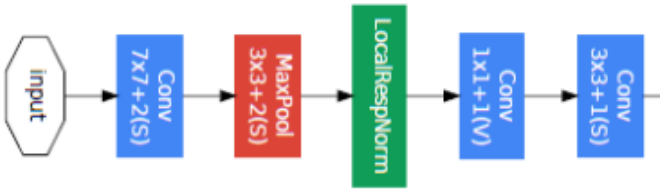


Figure 16: first three convolution layers used for our model

Figure 17 shows all 192 features that are generated by the first three layers of the googlenet model.

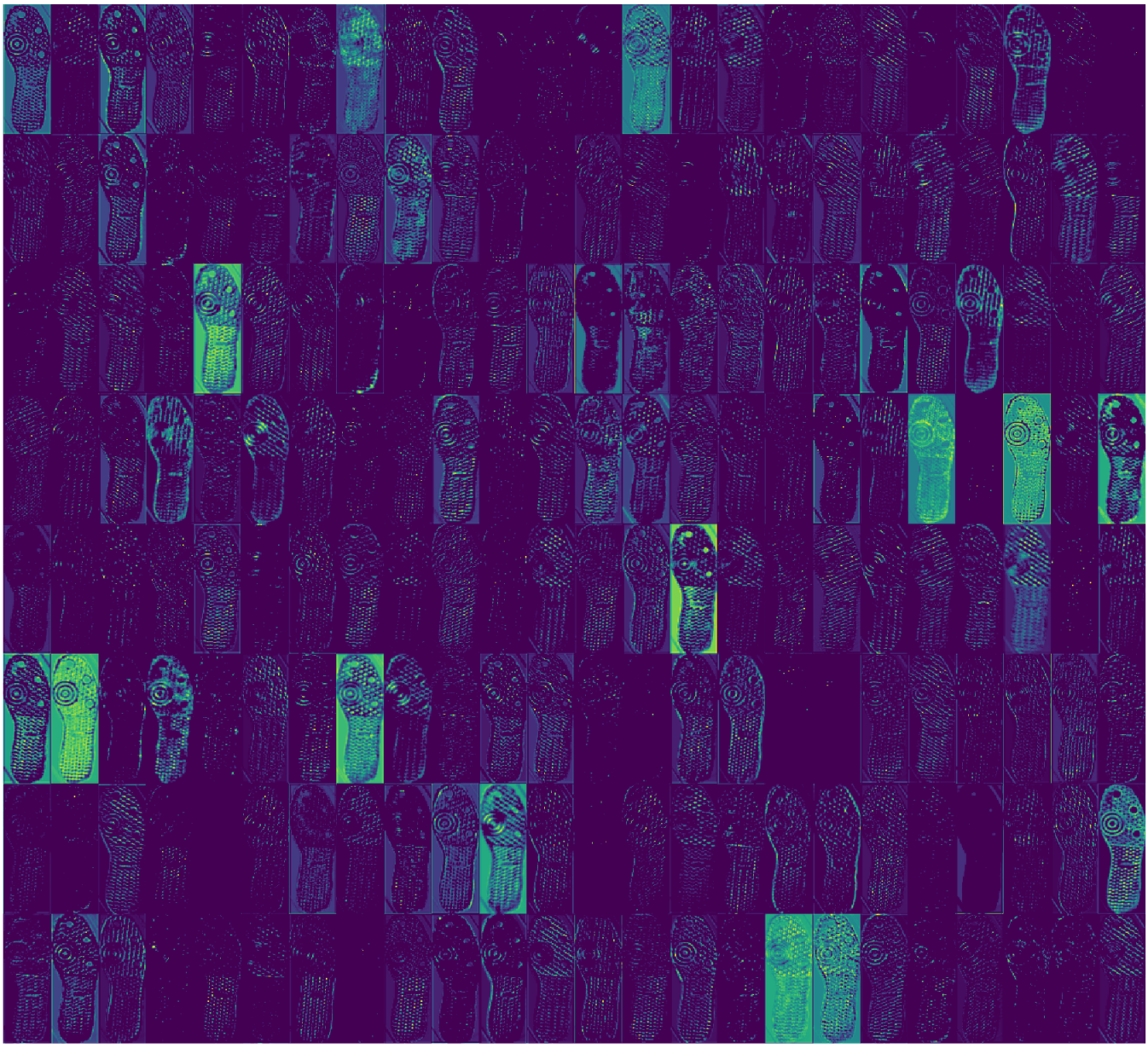


Figure 17: All 192 features (reference image 1041) generated by the first three layers of the googlenet model

4.2 Normalized cross correlation

Cross correlation between two 2-dimensional arrays is often used for template matching tasks, where the goal is finding specific features within an image. The cross correlation between two images/matrices yields another matrix which contains the cross correlation coefficients as its elements (Formula 9).

$$c(u,v) = \sum_{x,y} f(x,y)t(x-u, y-v) \tag{9}$$

Equation 9 depicts a term that describes the smilarity between the image and the feature with f as the image and t the feature positioned at u, v.

There are three main disadvantages to using this type of correlation:

1. If the image energy varies with position matching can fail. The correlation between a bright spot and the feature might overshadow the correlation of the region with the exact matching feature.
2. The range of c(u,v) depends on the size of the feature
3. c(u, v) is not invariant to changes in image amplitude.

The correlation coefficient overcomes those issues by normalilzing the image and the feature:

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\left\{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\right\}^{0.5}} \tag{10}$$

Where t is the mean of the feature and $f_{u,v}$ is the mean of the image region that is under the feature. Figure 18 visualizes the correlation surface that is computed by normalized cross correlation.

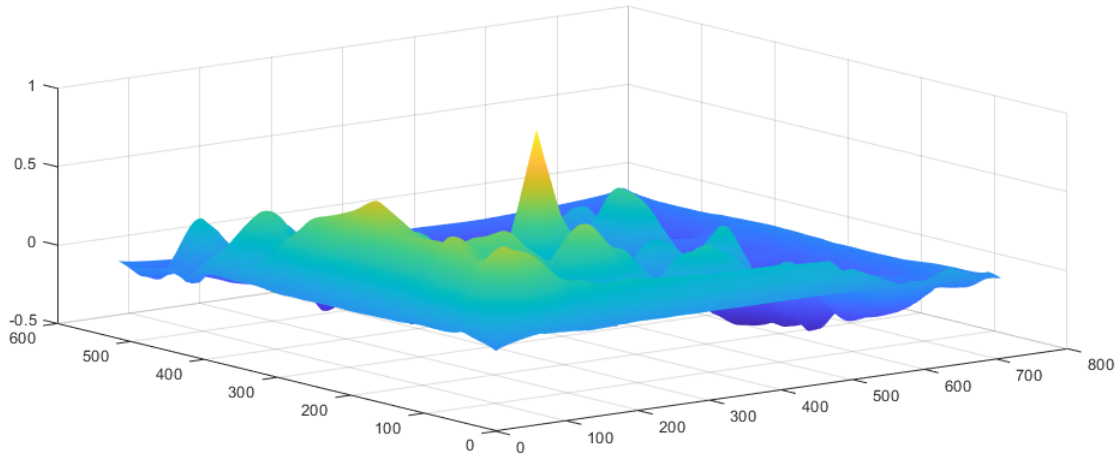


Figure 18: 3D-visualization of the correlation coefficient computed by normalized cross correlation

This correlation can be efficiently computed in the frequency domain using the fast fourier transformation. Unfortunately the normalized form of correlation that is prefered in template

matching does not have a simple frequency domain expression. For this reason the normalized cross-correlation has been computed in the spatial domain. Matlabs normxcorr2 function makes use of fast convolution and therefor enables a fast computation of the normalized cross-correlation. Unfortunately there is no such function for pytorch at this time. The computation of the normalized cross-correlation algorithm (Formula 10) on a reference image and its crop out, leads to a wall time of 26.8 seconds (see Figure 19). For the use of multiple feature channels in combination with NCC, this is therefore not a feasable option.

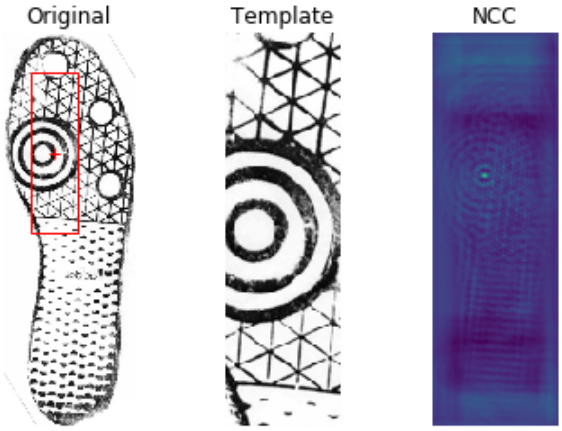


Figure 19: Left: Normalized cross correlation between a reference image and a crop out of it. Right: "Heat map" of the correlation coefficient

Figure 20 below shows the same procedure with an actual crime scene impression.

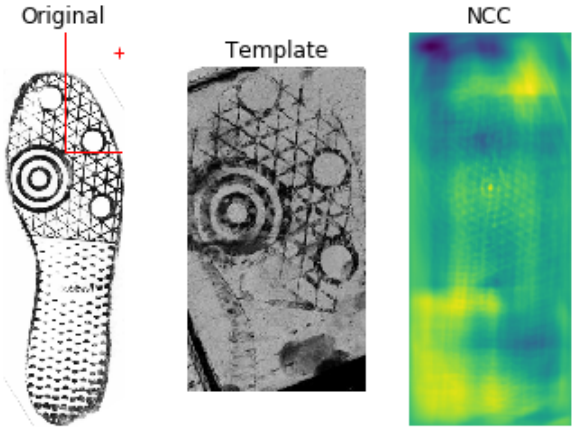


Figure 20: Left: Normalized cross correlation between a reference image and its corresponding track. Right: "Heat map" of the correlation coefficient

In 2019, Roger Bermudez-Chacon published an algorithm for normalized cross-correlation, that makes use of the pytorch conv2D function in order to enable a fast computation of cross-correlation between two images[4]. This significant time reduction in computation, allows for the use of multiple feature channels from a convolutional neural network.

[4]https://github.com/rogerberm/pytorch-ncc

4.3 Pre-processing

When using normalized cross correlation on the 192 features from the pre-trained convolutional neural network, the cross correlation heatmap shows two vertical and two horizontal artefact lines (see Figure 21).

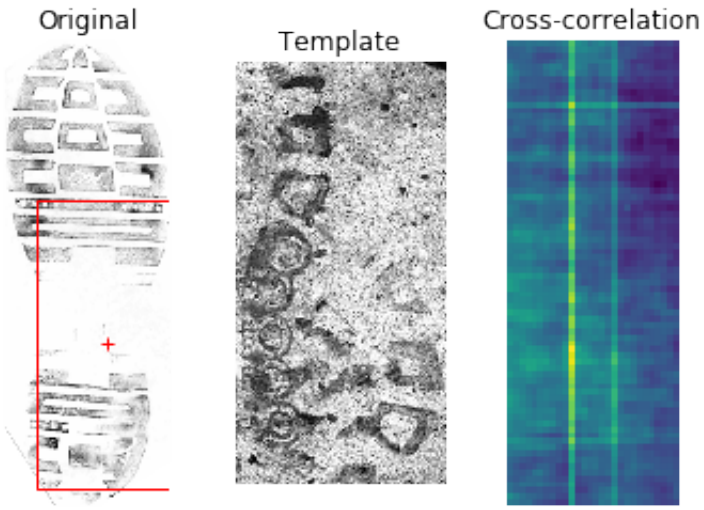


Figure 21: False classification example. The cross correlation heatmap shows two vertical and two horizontal artefact lines.

The features generated by the network exhibit lines on all edges on both, the reference image and the track image (see Figure 22). We strongly suspect that those artefacts seen in Figure 21 correlate with the lines seen in the features. We therefore cropped out those edges in order to get rid off the artefacts.

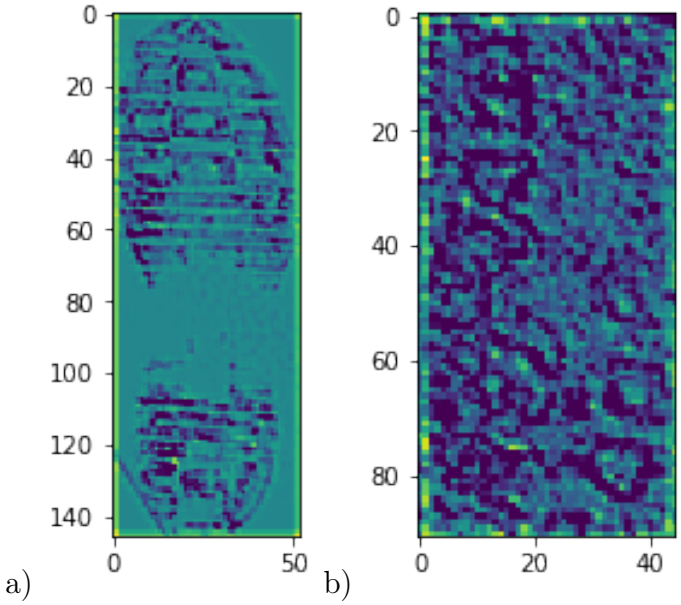


Figure 22: a) first feature from the reference image. b) first feature of the track image

4.4 Algorithimic efficiency

In order to make this algorithm applicable to many applications and large datasets, it is important to consider the algorithmic effenciency. In an effort to reduce the calculation time, we used different strides for the calculation of the correlation coefficients. Figure 23 shows the cumulative matching characteristics for different sizes of strides. The actual scores are not getting significantly worse by using higher strides sizes.

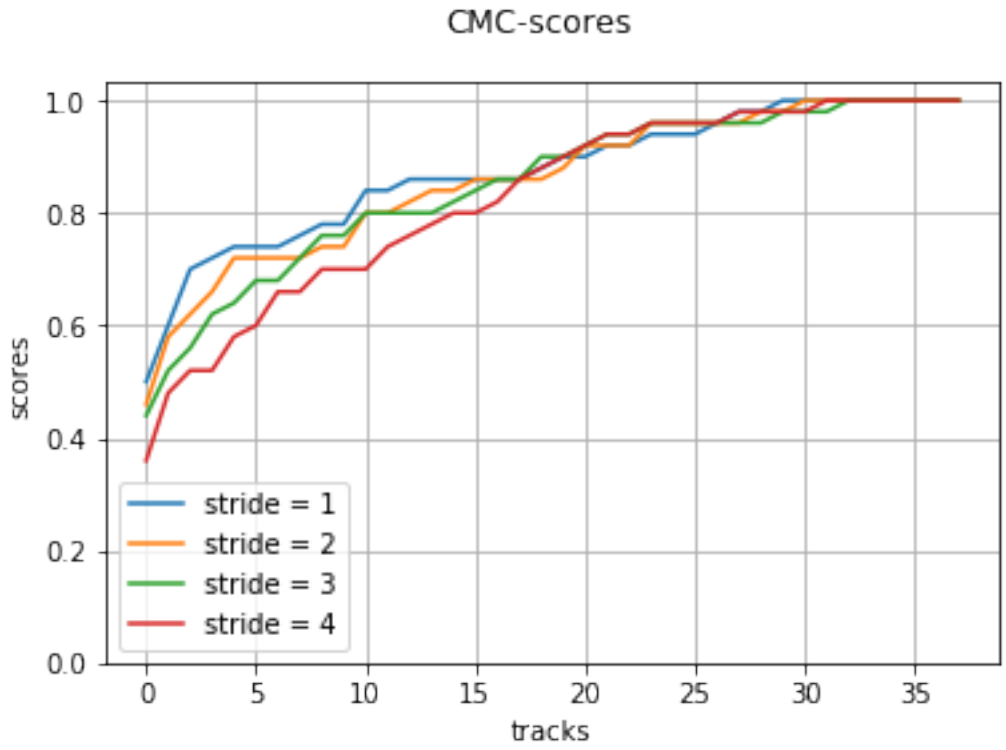


Figure 23: Comparison between CMC-scores using different strides. For this calculation a subset consisting of 38 track images and 50 reference images has been used.

In Table 1 we can see a big time reduction by using higher values for the stride, which makes this method usable for this type of matching tasks.

Strides	Time [GeForce GTX 1060]	Time [Tesla T4]
1	6032 sec	5346 sec
2	2404 sec	2103 sec
3	1902 sec	1218 sec
4	1601 sec	1110 sec

Table 1: Calculation time with different values for the stride. For the calculations the GeForce GTX 1060 and the Tesla T4, provided by google collaboratory, have been used.

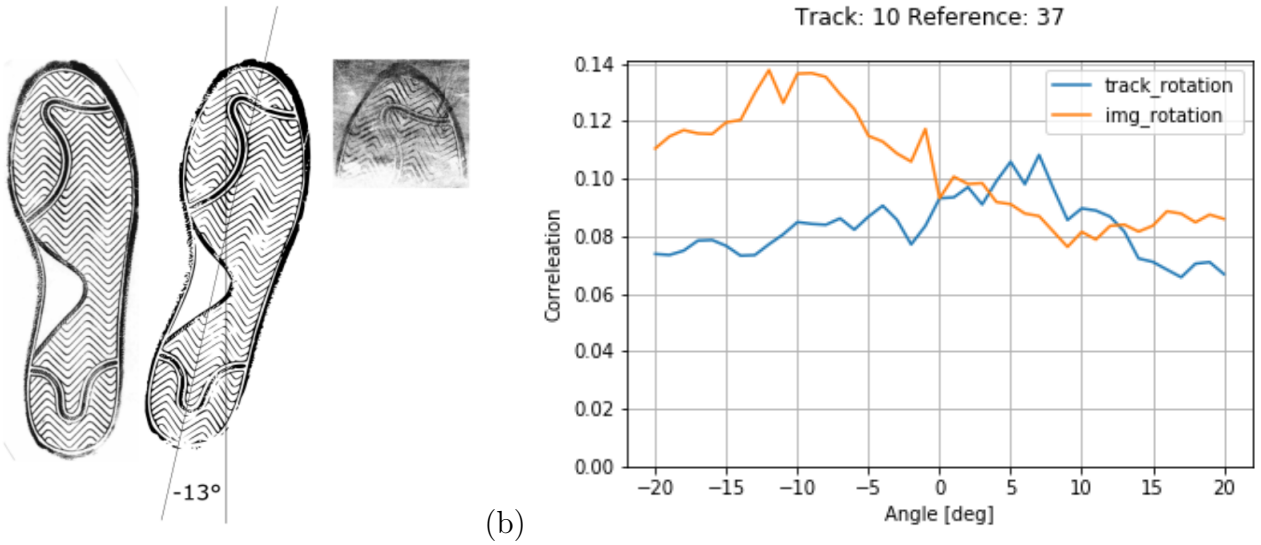
5 Results

5.1 Rotation

Due to the manual process of impression retrieval dicussed in chapter 2, a lot of crime scene prints in the FID-300 dataset are not vertically aligned as the reference images are. In order to retrieve more accurate values for the MCNCC-scores we first calculated the correlation for different states of rotation of the templates varying from either -10° to +10° or -20° to +20° with one degree steps inbetween. The same procedure was done with the reference images. Rotating the images leads to a loss of information, since part of the shoeprint can get out of the boundaries of the image. To avoid that, it is necessary to add sufficient padding increasing the width and hight of the image, which therefore leads to significantly higher calculation times.

5.1.1 Rotation of unaligned templates

To study the correlation coefficients in context of rotation, we first take a look at unaligned images and compare the resulting correlation-angle-diagrams with the actual rotations of the images. This case is depicted in Figure 24. We determined that the corresponding track image is twisted at an angle of roughly -13° compared to the reference impression.



(a) (b)

Figure 24: (a): Manual determination of rotation angle. (b) Correlation-rotation-diagram for track- and image rotation

For the rotation of the reference image we therefore anticipate a peak at around -13° and for the crime scene print at +13°. Figure 24 (b) shows that for reference rotation, we get the bahaviour we expected, but for track rotation it shows only a hint of improvement.

The same expirement was then carried out for a number of crime scene impressions to see if this behaviour is recurring.

5.1.2 Rotation of aligned templates

We also want to make sure the algorithm provides rational results in the case of already aligned images. That means that as long as both the reference-image and the track-image are fairly aligned, the maximum value of the correlation-angle diagram should be at around zero degrees. Figure 26 and Figure 26 each show a diagram depicting the correlation over a range of rotation angles in an interval [-10°, 10°].

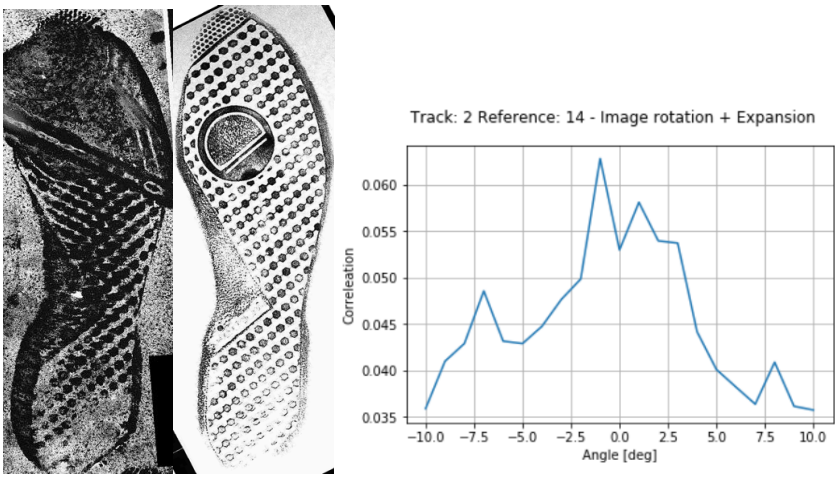


Figure 25: Correlation-rotation-diagram for track- and image rotation

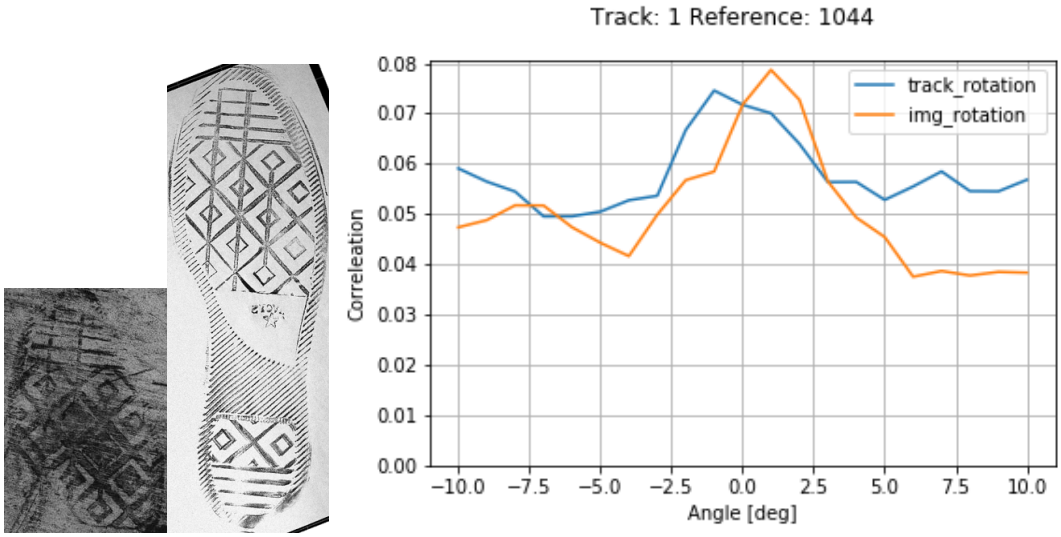


Figure 26: Correlation-rotation-diagram for track- and image rotation

In Figure 27 we compare the cumulative matching score[5] between the standard MCNCC and the same algorithm but with cropped out features (chaper 4.3) on a subset consisting of 50 reference images and their corresponding crime scene impressions. The scores do not show any significant improvement.

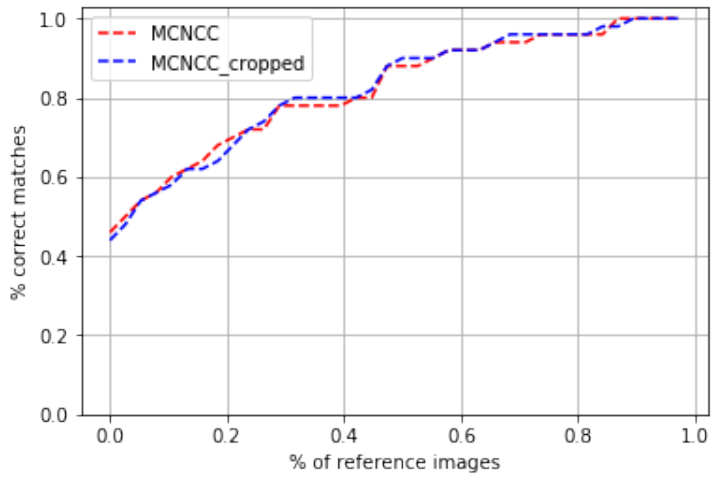


Figure 27: Comparison between MCNCC and MCNCC with cropped out features.

The same comparison has been carried out with rotation of the reference images, meaning that for each crime scene impression, we rotated each reference image in the subset from -10° to + 10° and used the maximum correlation coefficient for the MCNCC score matrix. This method showed slightly better results between the original features and the cropped features (see Figure 28).

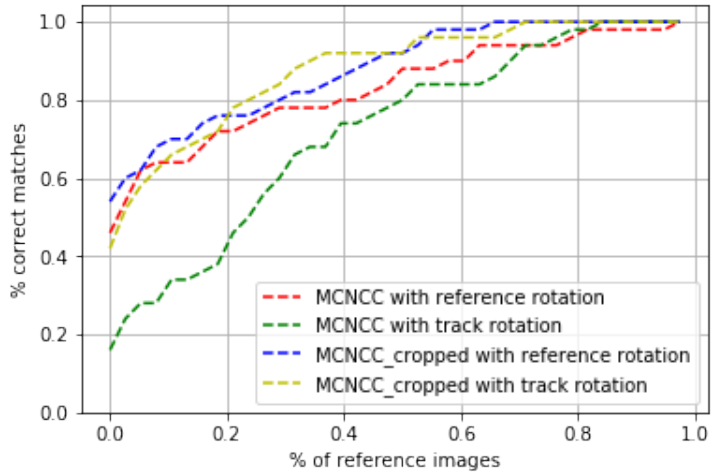


Figure 28: Comparison between MCNCC with reference rotation (red) and track rotation (green) and and MCNCC with cropped out features with reference rotation (blue) and reference rotation (yellow).

Figure 29 shows the rotation heatmaps for rotation angles in the interval from [-10°, 10°].

[5]https://github.com/ErikFaustmann/MCNCC/blob/master/cmc.py

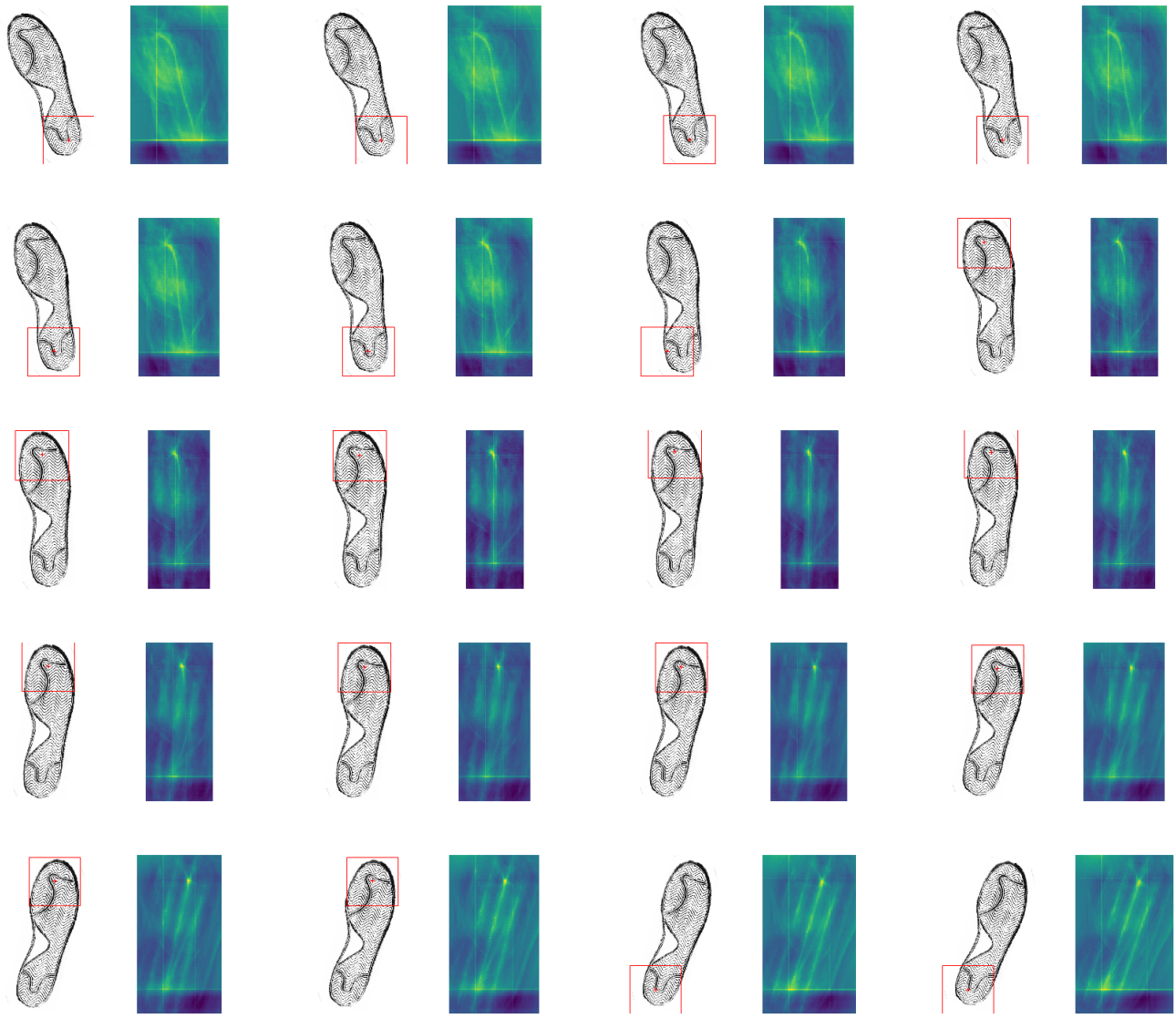


Figure 29: Correlation heatmaps for different rotation angles of the reference image, ranging from -19° to +19° in 2° steps.

5.2 Comparison

In Figure 30 we compare the results aquired by Kortilewski with our own results using MCNCC with cropped features and rotation of the reference images as discussed in chapter 5.1. The diagram shows that the cross-correlation with multiple features method was able to succesfully match more than 50% of the given dataset, but Kortilewskis results still show an overall better performance. It has to be considered that the diagram shows Kortilewskis scores using the whole FID-300 dataset, while for our curve we only used a subset of 50 crime scene impressions and the corresponding reference images due to limited computational capacity. This means that in Kortilewskis case there is a much higher chance for false matching errors. This condition has to be kept in mind while comparing there results. Using the whole dataset consisting of over a 1000 reference images might lead to a much worse matching performance for our method.

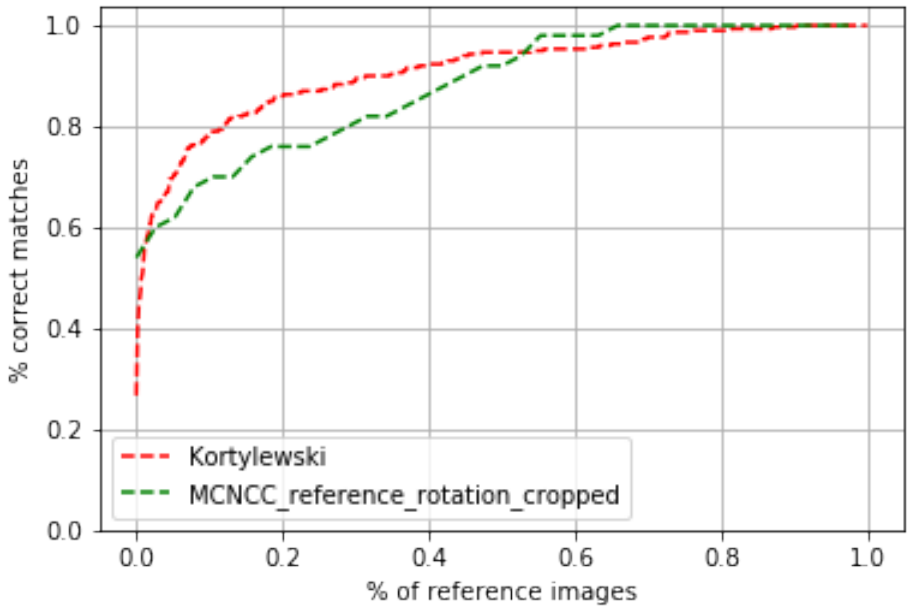


Figure 30: Comparison between Kortilewski's results and the results aquired from MCNCC

As discussed in chapter 4.4, we made an effort to reduce calculation time by using average pooling on the generated features in order to reduce the overall size of the feature maps. With this method we were able to significantly decrease the time of computation. For a small subset of 50 crime scene impressions and the corresponding track images and without using any rotations, the matching performance did not decrease in a major way. But using this configuration on the complete FID-dataset lead to a much worse result and took around 37 hours on a RTX 1060.

The computation of the cross-correlation matrix, which performance is represented in figure 30, took about 24 hours on the vienna scientific cluster[6] without average pooling and stride = 1 for the calculation of the correlation coefficient.

[6]https://vsc.ac.at/home/

6 Conclusion

The matching of images from different domains is made difficult by various reasons such as non-linear distortions, varying shoe sizes, and different rotational alignments for example. Based on the insights from [TR] we implemented a normalized cross correlation algorithm[7] that performs on a per-channel basis. This method already outperfoms the current state of the art, as published in [S1]. We implemented the version that uses uniform weights for each channel, which performs slighty worse than MCNCC with learned per-channel weights and learned projecton and per-channel weights (Siamese). In contrast to [TR] the model is not implemented in Matlab (which makes use of the normxcorr2 function for fast normalized correlation calculation). Instead, we used a python library called pytorch. We detected that feature maps generated from the first three convolution layers of the pre-trianed googlenet, show artefacts that have a negative influence on the matching algorithm. Cutting these artifacts out of our feature layers then improved the matching performance. We were also able show the positive effects when rotating the reference images in order to find the right alignment for a better correlation score. The problem with using the pytorch library is the increased time consumption of the algorithm. In order to tackle this problem, we tried average pooling for the feature maps in order to reduce their size and therefor decrease calculation time. For a small subset, this resulted in a slightly worse performance, but for larger sets it showed a significant decrease in matching.

[7] https://github.com/ErikFaustmann/MCNCC

7 Appendix

7.1 Normalized cross correlation - Example

Normalized cross correlation between two 2-dimensional arrays is often used for template matching tasks, where you want to find a smaller image within a larger one. The cross correlation between two images/matrices yields another matrix which contains the cross correlation coefficients as its elements.

$$image: \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \; template: \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix} \tag{11}$$

Depending on the size of those two original matrices, it can be necessary to add padding p to the larger image/matrix.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

From (12) we take the upper left 2x2 submatrix and substract the local mean $\mu = 0.25$ of this area. The same is done with the whole template matrix (global mean $\mu = 4$. This leads to (13):

$$\begin{bmatrix} -0.25 & -0.25 \\ -0.25 & 0.75 \end{bmatrix} \begin{bmatrix} -2 & -1 \\ 1 & 2 \end{bmatrix} \tag{13}$$

The next step is calculating the standard deviation of those matrices (5):

$$\sigma_{subwindow} = (-0.25)^2 + (-0.25)^2 + (-0.25)^2 + (0.75)^2 = 0.75 \tag{14}$$

$$\sigma_{template} = (-2)^2 + (-1)^2 + (1)^2 + (2)^2 = 10 \tag{15}$$

$$NCC = \frac{(-0.25)(-2) + (-0.25)(-1) + (-0.25)(+1) + (-0.25)(+2)}{\sqrt{0.75 \cdot 10}} \approx 0.73029 \tag{16}$$

The result of (16) would therefore be the first element of a 4x4 correlation matrix.

References

[S1] "Crime Scene Forensics: A Scientific Method Approach", Robert C Shaler

[TR] Bodziak W. J. Footwear Impression Evidence: Detection, Recovery, and Examination. 2nd ed. CRC Press-Taylor & Francis, Boca Raton, Florida, 2000, pp. 334, 347, 352, 413.

[KV] "Probabilistic Compositional Active Basis Models for Robust Pattern Recognition", Adam Kortylewski, Thomas Vetter

[L] "Fast Normalized Cross-Correlation", J. P. Lewis

[S] "Fast and High-Performance Template Matching Method"; Alexander Sibiryakov

[KSRF] "Cross-Domain Image Matching with Deep Feature Maps"; Balley Kong, James Supancic, Deva Ramanan, Charless C. Fowlkes

[BHJS] "Determining the significance of Outsole Wear Characteristics during the forensic examination of footwear impression evidence", William J. Bodziak, Lesley Hammer, G. Matt Johnson, Rodney Schenck, 2012

[SL] "Going Deeper with Convolutions", Christian Szegedy1 , Wei Liu2 , Yangqing Jia1 , Pierre Sermanet1 , Scott Reed3 , Dragomir Anguelov1 , Dumitru Erhan1 , Vincent Vanhoucke1 , Andrew Rabinovich4 1Google Inc. 2University of North Carolina, Chapel Hill 3University of Michigan, Ann Arbor 4Magic Leap Inc.