computer graphics & vision

Inffeldgasse 16/2, A-8010 Graz

www.icg.tugraz.at

TU Graz

Deep Restore

Using Image Segmentation and Image In-painting Techniques

Matthias Reiterer, BSc

Institute of Computer Graphics and Vision
Graz University of Technology, Austria

Seminar/Project Computer Vision
Advisor: Dipl.-Ing. Erich Kobler
Supervisor: Prof. Dr. Thomas Pock
Graz, October 31, 2018

Contact: Matthias Reiterer, BSc, matthias.reiterer@student.tugraz.at

Abstract

Analog movies, recorded on roll film, have the property that their image quality worsens statically either from usage or by time. To prevent this deterioration from happening, analog movies often get digitalized. The digitalization of analog movies has the advantage that during post-processing steps the image quality of previously analog movies can be improved - this is known as film restoration.

In this project, we will focus on such a post-processing step, namely the removing of impurities in single image frames. Based on previous research in the field of image restoration we came up with a step-wise solution: first, we detect the impurities of each image and generate a segmentation mask (image segmentation), and second, based on this segmentation mask we fill the impurities by using information from surrounding pixels (image in-painting).

Throughout this project, we mainly focus on modern machine learning approaches, especially convolutional neural networks (CNNs). For the image segmentation we test different architectures and regularization techniques to come up with a suitable architecture which gives the best results for the segmentation mask.

After we generated the segmentation mask we can perform image in-painting of the impure pixels. Again, we test different CNN-architectures and another method, image in-painting based on Perona-Malik-Diffusion.

*We evaluate our approaches on ground truth data given from HS-Art [**?**]. The results show that our method produces visually pleasing images.*

Keywords: *Computer Vision, Film Restoration, Machine Learning, Image Segmentation, Image In-painting*

Contents

1 Introduction

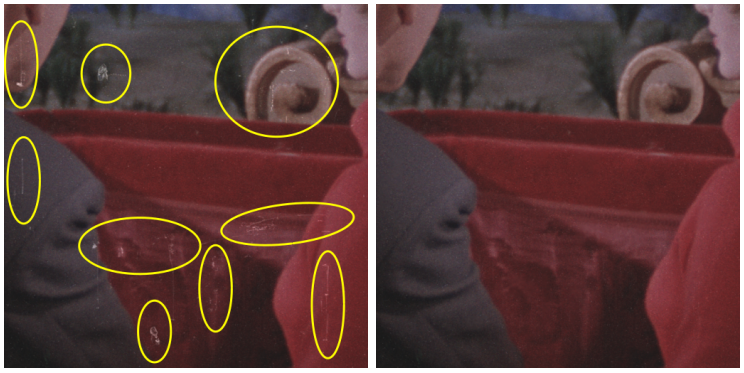
"Deep Restore" - a combination of the two terms "deep learning" and "image restore". As the name of this project suggests, we focus our considerations on image restoration based on deep learning methods. This project is done in cooperation with HS-Art [**?**], who specialize in fields of archive film restoration and preservation[1].

Image restoration describes the process of determining undistorted images from corrupted ones. It is used in a wide range of image and video processing applications and a basic method in computer vision. There exist many different approaches and an overview on some techniques can be found in Section **??**.

As Goodfellow et al. [**?**] state, deep learning is a part of machine learning methods based on feature learning and it is nowadays present in different fields like: computer vision, speech and audio recognition, bioinformatics and others. The learning of complicated tasks is based on the combination of many simpler ones and this hierarchy structure is deep, hence the name deep learning. There are different deep learning architectures like *e.g.* recurrent neural networks (RNNs), Boltzmann machines, convolutional neural networks (CNNs), *etc.* The learning of these architectures can either happen supervised, semi-supervised or unsupervised.

Since we focus on image restoration in this project, we decide to use CNNs to estimate undistorted images. CNNs are feed-forward neural networks and are typically applied to inputs with local structure, *i.e.* that consecutive and/or neighboring input features are more correlated than distant ones. Therefore, CNNs are mainly used in combination with images, video or audio. We choose to train our CNNs in a supervised manner. So, input images and ground-truth outputs combined are used to estimate the parameters of the networks.

As already mentioned, the goal of this project is to perform image restoration based on deep learning methods. Given a corrupted image (Figure **??**), we want to determine an undistorted image (Figure **??**).



(a) Impure image (b) Restored image

Figure 1: An example of image restoration. (a) shows an image with impurities (yellow circled regions), whereas (b) shows an undistorted/restored variant of (a).

By investigating our given images and by performing research, we conclude that we can categorize the impurities of corrupted images into: (a) dust, (b) noise, other damages (like (c) bent frames, (d) scratches, *etc.*) and for subsequent image frames (e) flicker. During this theses we focus on: dust, noise and other damages. An overview of the different impurities is given in Figure **??**, the images were taken from [**?**].

[1] All the images used in this project are provided by HS-Art [**?**].

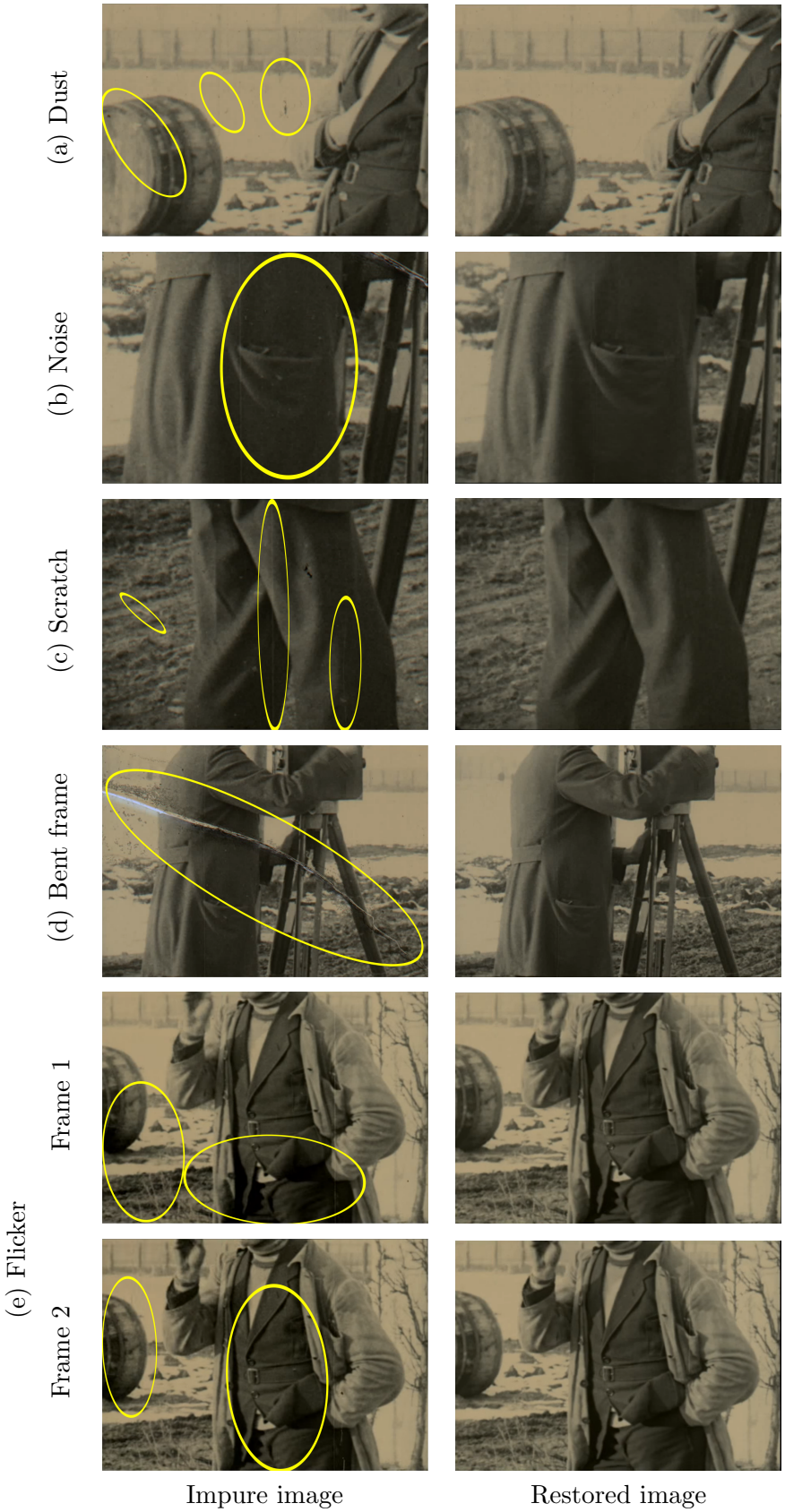
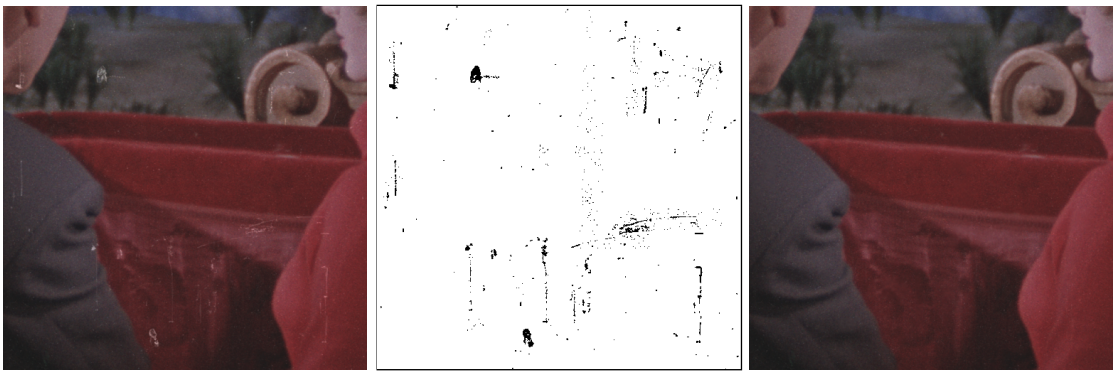


Figure 2: Different impurities and their restoration. For each of the different impurities (a) - (e) the corruption is highlighted in the left image (yellow circled regions) and the restored variant (right image) is shown.

We split the workflow of image restoration into two consecutive steps: image segmentation and image in-painting. In the image segmentation step we generate a segmentation mask based on the input image. Image segmentation is the task of assigning a label to each pixel. In our case we have two labels, one label representing impurities and the other clear pixels. The resulting segmentation mask highlights impurities which are then removed.

The next step is image in-painting. Image in-painting describes the process of filling in unknown pixels with information based on neighboring pixels. This is done by using the previously created segmentation mask and the input image.

This worklow can be seen in Figure **??**. In the first step we determine the segmantation mask (Figure **??**), where impurities are highlighted in white, from the input image (Figure **??**). Then, we perform image in-painting (Figure **??**) based on the segmentation mask (Figure **??**) and the input image (Figure **??**).



(a) Input image (b) Generated mask (c) Restored image

Figure 3: An example workflow of image restoration.

2 Related Work

Image restoration is a fundamental part of computer vision. There exist many different approaches and an overview on different techniques is given by Milanfar [**?**]. Most of the current work on image restoration focuses on developing methods for achieving visual great looking results. This is mostly done by the cost of computational efficiency. Chen et al. [**?**] propose a method based on diffusion processes, which also takes computational efficiency into account.

Since we decided to split up the task of image restoration into image segmentation and image in-painting, the following Sections give an overview of different image segmentation techniques (Section **??**) and image in-painting techniques (Section **??**).

2.1 Image Segmentation

Driven by a wide range of application, image segmentation received a significant amount of attention from the research community. Image segmentation is used in medical image processing for *e.g.* segmenting cancer cells, in various object detection tasks, like pedestrian detection, video surveillance and others.

Yuheng et al. [**?**] categorize segmentation algorithms into: (i) region-based threshold segmentation, (ii) regional growth segmentation, (iii) edge detection segmentation, (iv) segmentation based on clustering, and (v) segmentation based on supervised learning with CNNs. A newer approach from Qazanfari et al. [**?**] uses evolutionary computation that is inspired from human behavior to perform image segmentation.

We now focus on CNN based segmentation methods since this approach is most interesting for our project. The recent usage of CNNs for various computer vision tasks but also image segmentation led in partly groundbreaking results.

Long et al.[?] use fully convolutional networks to solve image segmentation. They adapt classification networks, such as VGG net [?] or GoogLeNet [?] into fully convolutional networks by transferring learned representations and combining semantic information from deep layers with appearance information from shallow layers.

In their work, Ronneberger et al. [?] propose a CNN architecture for image segmentation that consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. This U-Net architecture achieves good performance on different biomedical segmentation tasks and is also used in other areas. Huang et al. [?] introduce in their work a dense convolutional network (DenseNet), in which each layer is connected to every other layer in a feed-forward fashion. Feature-maps of preceding layers are used as input for the current layer. An advantage of this architecture is that the number of parameter gets reduced.

The recent work of Chen et al. [?] proposes new methods to further improve image segmentation with CNNs. They introduce convolution with upsampled filters to control the resolution at which feature responses are computed, a spatial pyramid pooling to segment object at multiple scales and to improve object localization, a combination of probabilistic graphical models and CNNs.

During our project we take inspiration from the work of Huang et al. [?] and Ronneberger et al. [?].

2.2 Image In-painting

Image in-painting is an ill-posed inverse problem with no well-defined solution. Its application reaches from removing text overlays or scratches, restoring lossy image transmission, object removal and others.

Guillemot et al. [?] give an overview of different image in-painting techniques and categorizes as follows: (i) in-painting using diffusion partial differential equations (PDEs), (ii) variational in-painting, and (iii) examplar-based methods. Another category is (iv) image in-painting with the help of deep learning and machine learning.

Diffusion based image in-painting, like the work of Boujena et al. [?], rely heavily on the famous work of Perona and Malik [?] and Weickert [?]. Image diffusion is based on PDEs, where diffusion describes a physical process. The crucial part to establish a visual appealing in-painting, is to incorporate a diffusion tensor into the equations (a detailed description can be found in Section **??**). The work of Getreuer [?] and Neri et al. [?] is based on total variation. The use slight variations of the total variation model for improved results.

Related works in the machine learning sectors are *e.g.* the work of Köhler et al. [?] and Gupta et al. [?]. Köhler et al. use a CNN to perform image in-painting. They state that with an additional segmentation mask, highlighting missing pixels, the results of the in-painting can be improved. Gupta et al. use both CNN and RNN approaches. Another recent trend in machine learning, that is popular in image in-painting, is the usage of an encoder-decoder structure or Generative adversarial network (GANs). Burlin et al. [?], Yeh et al. [?] and van Noord et al. [?] all use this structure to perform state-of-the-art image in-painting.

3 Image Segmentation

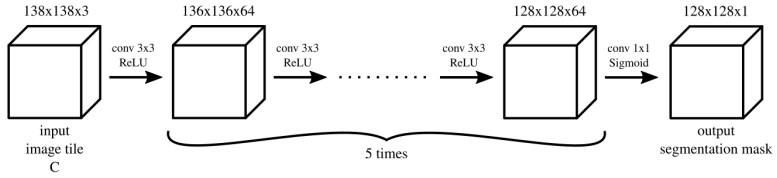
In this Section follows a description of our methods and approaches for detecting impurities by image segmentation. As already mentioned we focus on a machine learning approach namely CNNs. Section **??** gives an overview of architectures that use just the current image to perform

the image segmentation. Section **??** shows architectures that use additional information, the previous and next image, to perform image segmentation of the current image. And Section **??** lists the approach of using additional warped instances of the previous and next image.

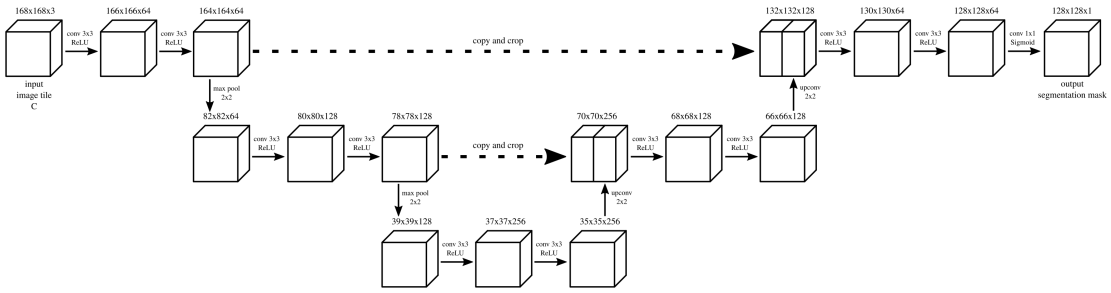
It should be mentioned that if not stated otherwise, we use ReLU activation functions between layers and a sigmoid output activation function to perform image segmentation, which is in general a classification task. We also apply regularization techniques that are described in Section **??**. We choose the cross-entropy-error (CEE) for computing the loss/error between predicted and ground-truth segmentation masks.

3.1 Single Input Image

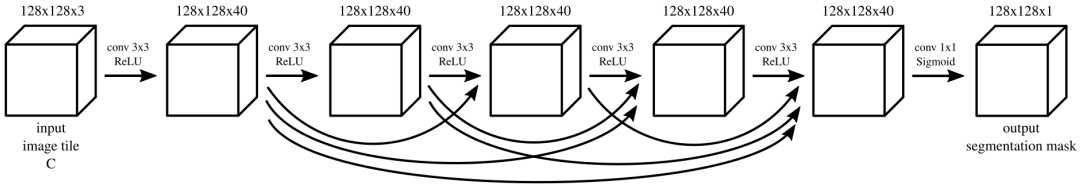
Our first approach is to train simple architectures with few parameters to perform image segmentation (workflow: from Figure **??** to Figure **??**). We train all following networks on pairs of: single images + ground-truth masks. The different networks (Figure **??**) are: (a) a simple one, consisting of five blocks, (b) a network based on the work of Ronneberger et al. [**?**], consisting of three layers and that uses a combination of max-pooling and de-convolution, and (c) a network based on the work of Huang et al. [**?**] that consists of five sub-blocks. The results of the different architectures are listed in Section **??**.



(a) A "simple"-network with five blocks. An additional padding of size 5 was added to each 128×128 patch because valid-convolution was used.



(b) An U-Net-network with three layers. An additional padding of size 15 was added to each 128×128 patch because valid convolution was used.



(c) A densenet-network with five sub-blocks, this is later referred to as a single densenet block.

Figure 4: Different CNN architectures for image segmentation. The "C" in each Figure (a) - (c), in "image tile C", stands for current and means that we just use information from the current image.

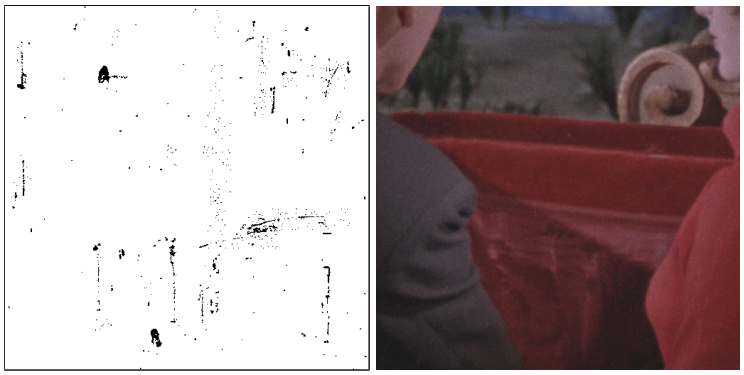
Using just a single input image we achieved bad results (see Section **??**). Thus, we decided to add additional information to our input, see Section **??**.

3.2 Multiple Input Images

This Section gives an overview of architectures that use additional input information to perform image segmentation (workflow: from Figure **??** and Figure **??** to Figure **??**). We now also use the previous and next image to perform image segmentation on the current frame, see Figure **??**. In Figure **??** we can also see that impurities of the current frame (Figure **??**) are mostly not visible in the previous (Figure **??**) and next frame (Figure **??**).

(a) Previous image (b) Input image (c) Next image

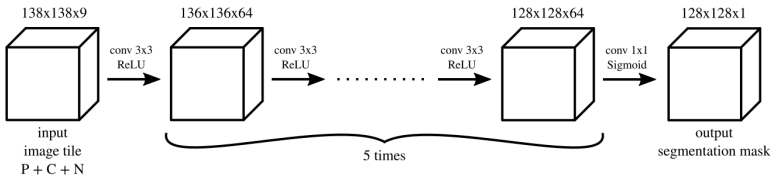
Figure 5: An example of using additional information for image restoration by the usage of previous (a), current input (b) and next image (c).



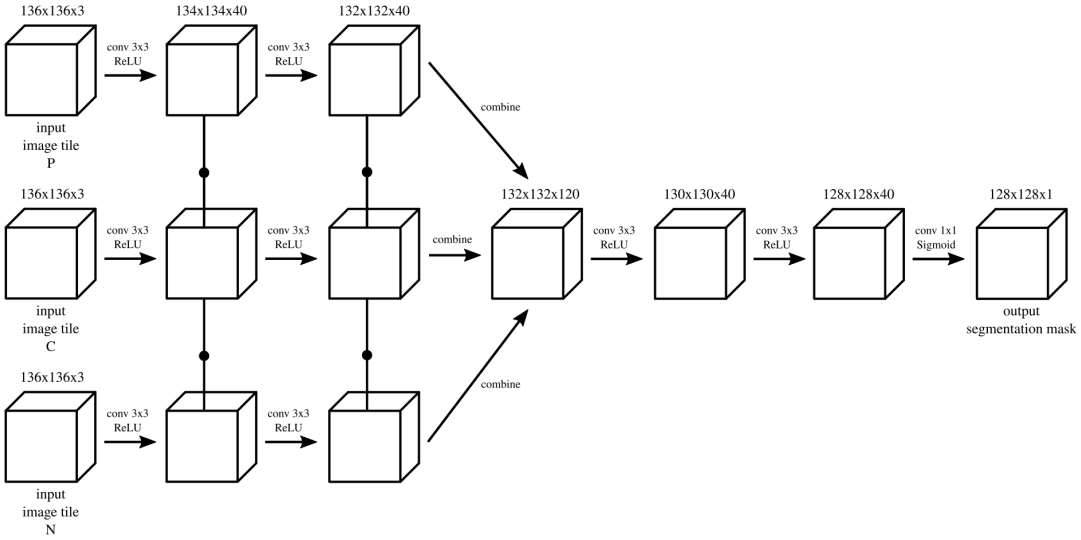
(a) Generated mask (b) Output image

Figure 6: An example workflow of image restoration using additional information (previous, current and next image, *e.g.* Figure **??**) for generating the segmentation mask (**??**) and inpainting (**??**).

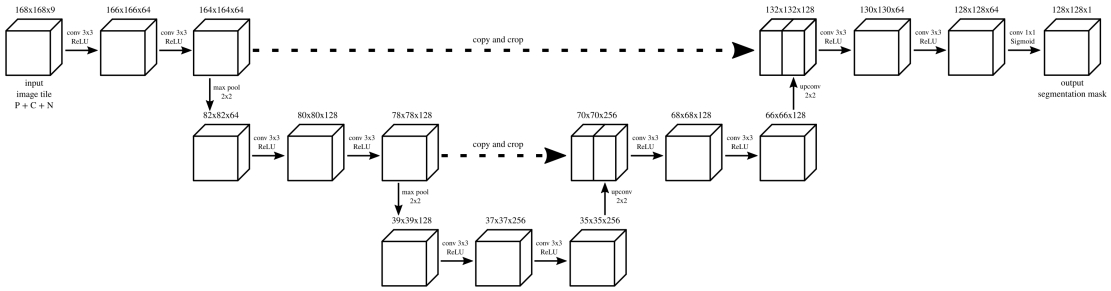
The different networks (Figure **??**) tested are: (a) an early combine network consisting of five blocks. We name this network "early combine" since we combine the information of all images in the beginning. (b) A late combine network that consists of two shared blocks (the line with the filled circle in Figure **??** and **??** means that weights and biases are shared for this block). We name this network "late combine" since we combine the information of all images later. (c) A network based on the work of Ronneberger et al. [**?**], consisting of three layers and that uses a combination of max-pooling and de-convolution. (d) A network based on the work of Huang et al. [**?**] that consists of five sub-blocks. (e) A network, again based on the work of Huang et al. [**?**], that consists of an upper path and a lower path. The upper path consists of two densenet blocks and the lower path of two shared densenet block. We combine the two paths via concatenating to achieve a final segmentation mask. The idea behind this concept is that the upper and lower path may learn different features that combined achieve better results. (f) A bottleneck network consisting of five blocks. We name this network "bottleneck" because of its structure to reduce the features via a 1×1 convolution. This is done to reduce the parameters of the network. And, (g) a network based on the work of Huang et al. [**?**] and Ronneberger et al. [**?**]. It consists of an upper path and a lower path. The upper path consists of two densenet blocks and the lower path of an U-Net block with three layers. We subtract the intermediate results of the two paths to achieve a final segmentation mask. The idea behind this concept is the same as for the "densenet combine"-network (e). The results of the different architectures are listed in Section **??**.



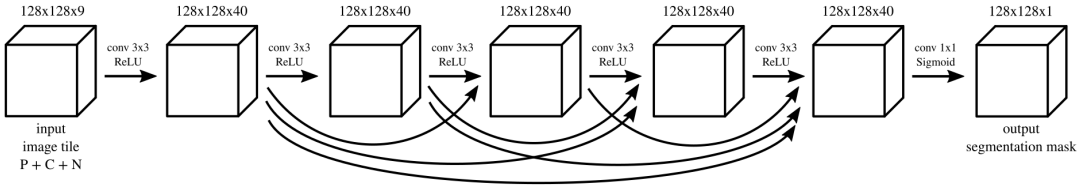
(a) A "simple"-network with five blocks. An additional padding of size 5 was added to each 128×128 patch because valid-convolution was used.



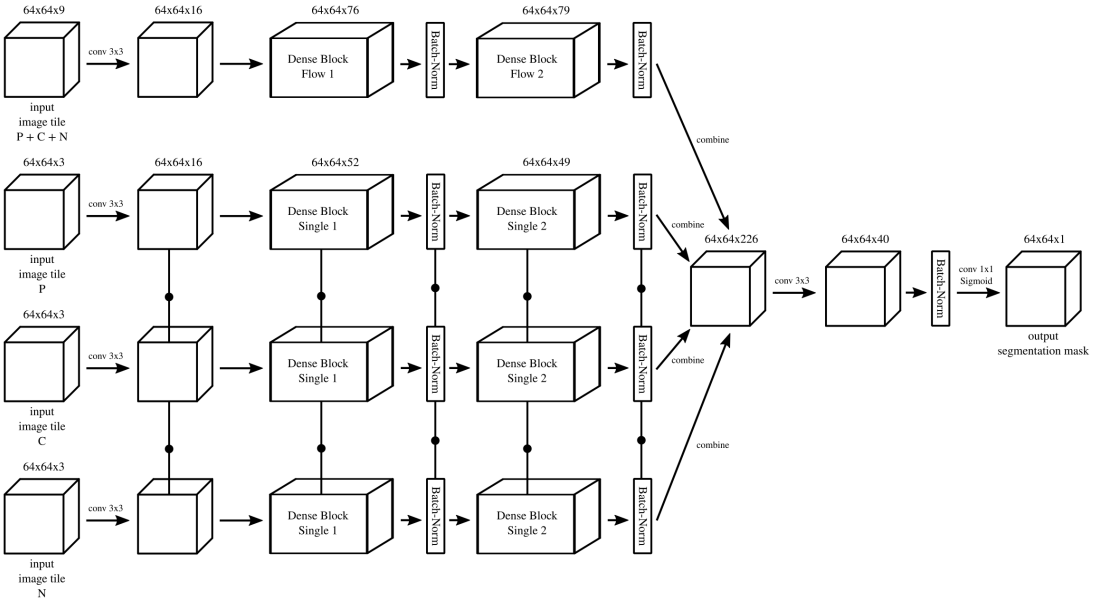
(b) A "late combine"-network with two shared blocks. An additional padding of size 4 was added to each 128×128 patch because-valid convolution was used.



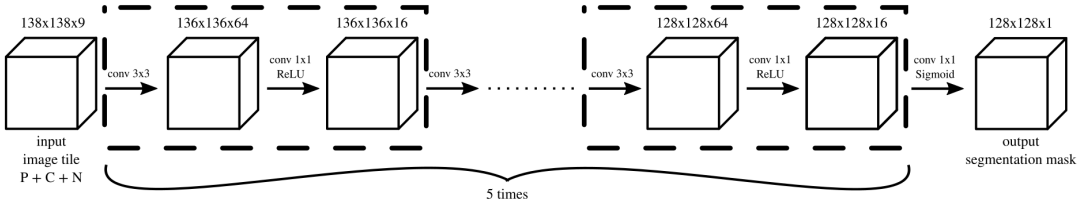
(c) An U-Net-network with three layers. An additional padding of size 15 was added to each 128×128 patch because valid convolution was used.



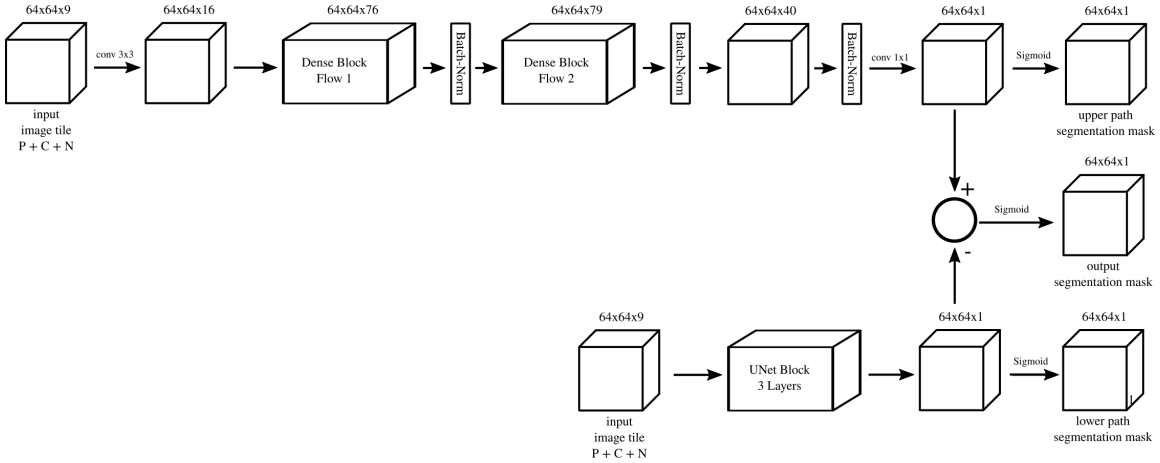
(d) A densenet-network with five sub-blocks, this is later referred to as a single densenet block.



(e) A "densenet combine"-network. The upper path consists of two densenet blocks and the lower path of two shared denesnet block. The two paths get combined via stacking to achieve a final segmentation mask. We train this architecture on 64×64 sized patches.



(f) A "bottleneck"-network with five blocks. An additional padding of size 5 was added to each 128×128 patch because valid convolution was used.



(g) A "densenet combine 2"-network. The upper path consists of two densenet blocks and the lower path of an U-Net block with three layers. The intermediate results of the two paths get subtracted to achieve a final segmentation mask. We train this architecture on 64×64 sized patches.

Figure 7: Different CNN architectures for image segmentation. The "P" and "N" in each Figure (a) - (g), in "image tile P + C + N", stands for previous and next, and means that we use additional information from previous and next image as well.

3.3 Multiple Warped Input Images

Based on the results of using additional information from previous and next images (Section **??**), we decide to test warped instances of the previous and next image. The results for Section **??** suggest that for sequences with a large optical flow, the tested architectures do not just estimate the impurities but also detect the optical movement. Therefore, we warp the previous and next image to match the current image to suppress the optical flow. For this Section, we use the same architectures as in Section **??**.

Figure **??** shows an example of the input images as we use them in Section **??**. In Figure **??** one can see warped instances of the previous and next image. For this example, the warping can be best seen by comparing the bars of the roof next to the head.

(a) Previous image (b) Input image (c) Next image

Figure 8: Example input images as in Section **??**.

(a) Warped previous image (b) Input image (c) Warped next image

Figure 9: Example input images for this Section. (a) is a warped instance of Figure **??** and (c) a warped instance of Figure **??**.

We train all the CNN architectures once for the warped image instances and once without warping to see if there is a difference. The results are listed in Section **??**.

3.4 Regularization Techniques

Techniques to prevent deep learning algorithms from overfitting are called regularizer. The following Sections **??** - **??** give a description of the different methods used in our CNN models (Section **??** - **??** and **??**).

3.4.1 Early Stopping

By investigating the trend of test and training error over iterations of deep learning algorithms (Figure **??**), one can observe that the training error decreases over iterates, but, the test error first decreases and then increases.

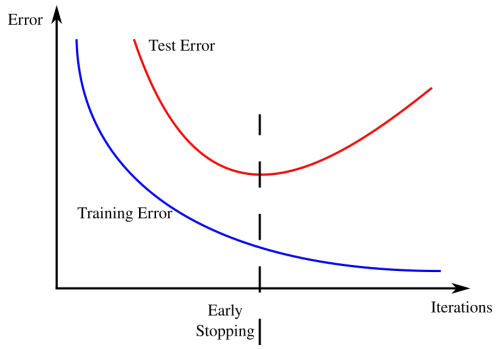


Figure 10: Trend of training and test error over iterates of deep learning algorithms. The training error is increasing. The test error decreases first and then increases.

Early stopping now works as follows: we monitor the error on the test set and store the model parameters every time we get a smaller error, after we determined with the number of iterations, we restore our model parameters with the lowest error on the test set. Early stopping acts as a regularizer because with increasing iterations the parameters grow and the effective model capacity grows.

3.4.2 Dropout

As Hinton et al. [**?**] state, the idea behind dropout is that neurons can not fully rely on the output of other neurons. So, co-specialization is not possible.

With dropout, a neuron/unit is dropped during training with a probability $1-p$, where p is the so called "keep probability". To drop a unit means, that both in the forward and backward-pass, the unit is ignored (output is set to 0). For our models we choose a keep probability p of 0.85.

3.4.3 L_2-Regularization (Weight Decay)

L_2-Regularization is a form of adding a parameter norm penalty $\Omega(\mathbf{W})$ to the error E, Eq. (**??**), with $\mathbf{W}$ being the weights of our model (the biases are not regularized):

$$\tilde{E}(\mathbf{W}, D) = E(\mathbf{W}, D) + \alpha\Omega(\mathbf{W})\,, \tag{1}$$

with

$$\Omega(\mathbf{W}) = \frac{1}{2}||\mathbf{w}||_2^2 \tag{2}$$

and $\mathbf{w}$ as the vectorized weights. This added L_2-Regularization , Eq.(**??**), leads to a weight decay. For the parameter α, we choose a value of 10^{-3} for our models.

3.4.4 Input Normalization

Input normalization is a technique that can speed up learning. We normalize each input feature to have zero-mean and unit-variance, like in Eq. (**??**):

$$\tilde{x}_i^{(n)} = \frac{x_i^{(n)} - \bar{x}_i}{\sigma_{x_i}}\,, \tag{3}$$

with $\bar{x}_i$ as the mean value of the i^{th}-feature of x, x_i,

$$\bar{x}_i = \frac{1}{N}\sum_{j=1}^{N} x_i^{(j)}\,, \tag{4}$$

and σ_{x_i} as the standard deviation of the i^{th}-feature of x, x_i,

$$\sigma_{x_i} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_i^{(j)} - \bar{x}_i)^2}\,. \tag{5}$$

3.4.5 Batch Normalization

The output of a layer can be interpreted as the input of the next layer. In batch normalization, this normalization (like in Section **??**) is done to the output of each layer for the current batch. As Ioffe et al. [**?**] mention, this technique leads to better learning dynamics. Batch normalization can be seen as an intermediate layer that linearly transforms the output of a layer.

4 Image In-painting

In this Section follows a description of our methods and approaches for the removing of impurities by image in-painting. Again, we focus on machine learning approaches namely CNNs but compare these against a diffusion based image in-painting. Section **??** gives an overview CNN architectures that are trained for a single color channel. Section **??** gives an theoretical overview on diffusion based image in-painting, as well as the exact used approach. The results of the different image in-painting approaches can be seen in Section **??**.

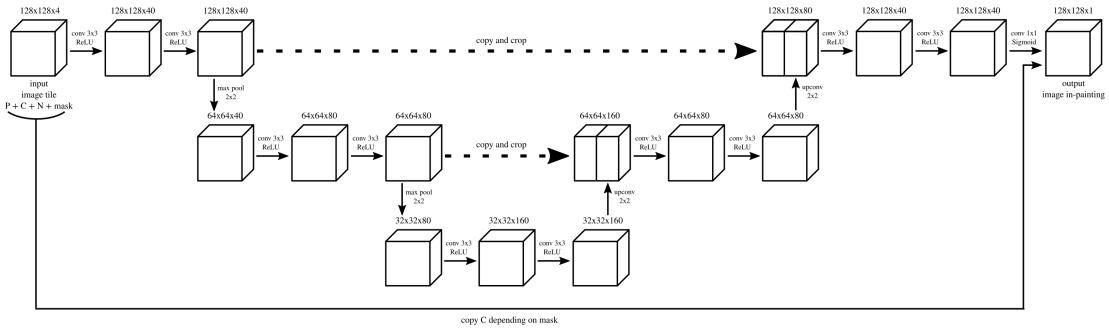
4.1 CNN

Based on the results of Section **??** we decide to use an U-Net approach and a densenet approach. Additionally, as seen in Section **??**, we use information from previous and next images as well. We train all following networks on quintuples of: previous - current - next images - masks + ground-truth in-painting.

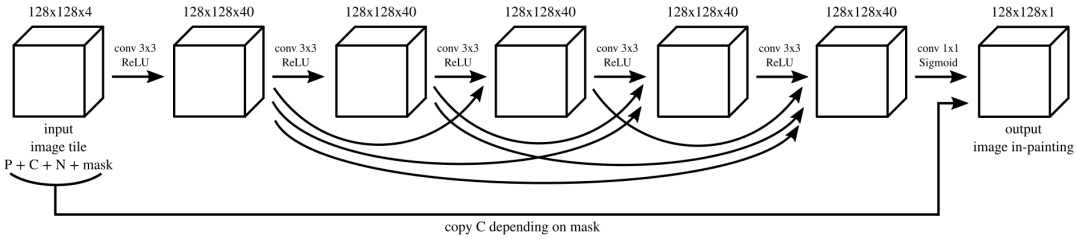
It should be mentioned that we use ReLU activation functions between layers and a sigmoid output activation function to perform image in-painting. We also apply regularization techniques, namely early stopping (Section **??**), dropout (Section **??**) and L_2-regularization (Section **??**). We choose the L_2-loss for computing the loss/error between predicted and ground-truth in-painting but just over masked pixels. And, we copy valid input pixel values into the predicted in-painting.

Since we train on a single color channel, we also have to perform the testing on each color channel separately and combine the intermediate results.

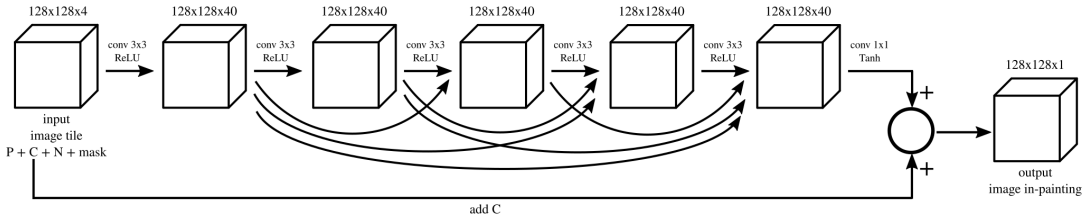
The different networks (Figure **??**) tested are: (a) a network based on the work of Ronneberger et al. [**?**] consisting of three layers, a (b) a network based on the work of Huang et al. [**?**] that consists of five sub-blocks, and (c) a network with two paths, where we add the input image in the end. This network is also based on the work of Huang et al. [**?**] and for this network we use a Tanh output activation function.



(a) An U-Net-network for image in-painting with three layers.



(b) A densenet-network for image in-painting with five sub-blocks.



(c) A densenet-network for image in-painting with five sub-blocks and two paths.

Figure 11: Different image in-painting CNN architectures.

4.2 Perona-Malik In-painting

The second image in-painting approach is based on a modified version of coherence-enhancing image diffusion. The theoretical basis are after Perona and Malik [**?**] or Weickert [**?**]. Now follows a description of the used method for one color channel.

Let $u : \Omega \to \mathbb{R}$ be the image function, with $\Omega \subset \mathbb{R}^2$ the image domain. The basic diffusion equation is given by:

$$\frac{\partial u}{\partial t} = \operatorname{div}(\nabla u)\,. \tag{6}$$

Eq. (**??**) is a PDE and is used to describe *isotropic* diffusion. To increase the modeling accuracy, a so-called diffusion tensor D is introduced into Eq. (**??**):

$$\frac{\partial u}{\partial t} = \operatorname{div}(D\nabla u)\,. \tag{7}$$

Eq. (**??**) describes *anisotropic* diffusion and the diffusion tensor D models rate of diffusion in different directions. If D is the identity matrix, we again obtain *isotropic* diffusion.

Coherence-enhancing diffusion leads the diffusion along coherent regions, similar structured regions, in our case edge directions. We achieve this, by using a structure tensor S (2×2 matrix). The structure tensor is able to give for every pixel information about its surrounding structure, and is calculated via:

$$S = G_\sigma * \begin{bmatrix} \tilde{u}_x^2 & \tilde{u}_x\tilde{u}_y \\ \tilde{u}_x\tilde{u}_y & \tilde{u}_y^2 \end{bmatrix}, \tag{8}$$

with subscripts denoting directional derivatives. G_σ is a Gaussian smoothing kernel with σ_t, and $\tilde{u}$ is a Gauss-filtered version of u with σ_g. The eigenvalues μ_1, μ_2 and eigenvectors ν_1, ν_2 of the structure tensor S give information about the local structure of the image, with the eigenvalues determining the magnitude of the grayvalue variation, the corresponding eigenvector giving the direction of the grayvalue variation:

1. μ_1, μ_2 both small: flat region,
2. $\mu_1 \gg \mu_2$ or vice-versa: strong grayvalue variation in one direction, *i.e.* image edge,
3. μ_1, μ_2 both large: strong grayvalue variation in two directions,*i.e.* image corner,

with this information we will guide the diffusion process.

The diffusion tensor D is calculated as:

$$D = \begin{bmatrix} \nu_1 & \nu_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \nu_1^T \\ \nu_2^T \end{bmatrix}, \tag{9}$$

using a small constant $\alpha > 0$, λ_1 and λ_2 are given as

$$\begin{cases} \lambda_1 = \alpha \\ \lambda_2 = \alpha + (1-\alpha)(1 - g(|\mu_1 - \mu_2|)) \end{cases} \tag{10}$$

and $g(s) = e^{-\frac{s^2}{2\gamma^2}}$ with a parameter γ.

We solve Eq. (**??**) with a semi-implicit approach. Since we are in the discrete world, an image of size $M \times N$ is represented as a vector $U \in \mathbb{R}^{MN}$, the time derivative $\frac{\partial u}{\partial t}$ and $K \in \mathbb{R}^{2MN \times MN}$ the equivalent to ∇ approximated by finite differences, $D(U^t) \in \mathbb{R}^{2MN \times 2MN}$ the diffusion tensor, τ the time discretisation step and since div $= -\nabla^T$ we then can write Eq. (**??**) as

$$\frac{U^{t+1} - U^t}{\tau} = -K^T D(U^t) K U^{t+1} \,. \tag{11}$$

Re-writting Eq. (**??**) and the introduction of a diagonal mask matrix $Q \in \mathbb{R}^{MN \times MN}$ that highlights pixels for in-painting, and $\hat{Q}$ the inverse of Q for highlighting preserved regions and solving for U^{t+1} leads to the image in-painting equation for one channel:

$$\begin{aligned} Q(U^{t+1} - U^t) + \hat{Q}\tau K^T D(U^t) K U^{t+1} &= 0 \\ QU^{t+1} - QU^t + \hat{Q}\tau K^T D(U^t) K U^{t+1} &= 0 \\ (Q + \hat{Q}\tau K^T D(U^t) K) U^{t+1} &= QU^t \\ U^{t+1} &= (Q + \hat{Q}\tau K^T D(U^t) K)^{-1} Q U^t \,. \end{aligned} \tag{12}$$

The extension of this single channel image in-painting to color images is done by applying the diffusion for each channel. To avoid artifacts many different approaches exist. As Åström et al. [**?**] state one could use a different color space, like HSV instead of RGB, use one diffusion tensor for each channel that is computed as the weighted average sum of the structure tensors for each color channel, use a different structure tensor for each channel, *etc.* We choose the approach of using the weighted sum over all channels for the structure tensor and use this for each channel.

5 Evaluation

We evaluate our different approaches with ground-truth images from HS-Art [**?**]. A description of this dataset follows in Section **??**. The needed implementation details for the evaluation are stated in Section **??** and in Section **??** our results are presented.

5.1 Dataset

HS-Art [**?**] provides us with a total of 173 image sequences. Per sequence we get an input image (current image) that is corrupted, the previous image in the sequence, the next image in the sequence, the ground-truth segmentation mask, highlighting impurities in the current image as well as ground truth image in-painting results of the current image. Each image is of size 512×512. For 77 out of these 173 image sequences we additionally have warped instances of the previous and next image, these sequences are used to evaluate Section **??**. For Section **??**, **??** and **??** we use the other 96 image sequences.

For training and evaluation of our CNN approaches we split the sequences into 80% training sequences and 20% test sequences. The training is done on patches of size 128×128, unless otherwise stated, and we train on patches were atleast 1% and at most 80% pixels are impure.

For the image segmentation task of using multiple input images (Section **??**) we further categorize our test set into sets that have: (i) ground-truth masks with large areas, (ii) ground-truth masks with small areas, (iii) sequences with large flow, and (iv) sequences with small flow.

We also perform dataset augmentation to get more data for training. Here we perform mirroring of the images and masks as well as a flow reversal by exchanging previous and next image.

5.2 Implementation Details

The different parameters used for our CNN approaches are listed either in Section **??** or in the corresponding model definitions. Additionally, we use the ADAM-optimizer with a learning rate of 10^{-3} to train other models.

For the Perona-Malik image in-painting approach (Section **??**) we use the following values for the parameters: $\alpha = 0.005$, $\gamma = 0.001$, $\tau = 5$, end time $= 15$, $\sigma_g = 0.7$ and $\sigma_t = 1.5$.

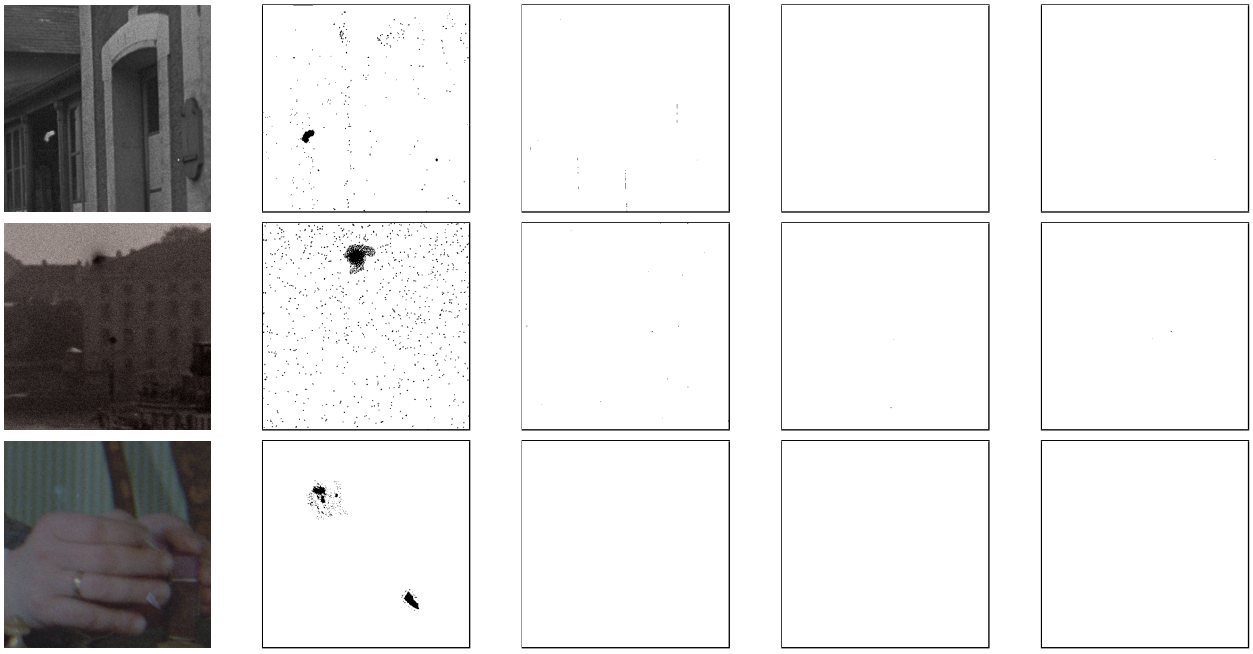
The project was realized in python and we used the Tensorflow API to implement our CNN approaches. The training and testing was done on a NVidia Geforce Titan X.

5.3 Results for Image Segmentation

In the following Sections we list the results of image segmentation using a single image (Section **??**), additional information from previous and next images (Section **??**) and additional information by using warped instances of previous and next images (Section **??**).

5.3.1 Single Input Image

The results in Figure **??** suggest that the information given in just a single current image is not enough for a sufficient learning and estimating of the segmentation mask. Therefore, we add additional information by using the previous and next image frames for all additional approaches.



(a) Input (b) GT (c) Simple (d) U-Net (e) Densenet

Figure 12: Sample results of single image segmentation for Simple (Figure **??**), U-Net ((Figure **??**) and Densenet (Figure **??**).

5.3.2 Multiple Input Images

Following prior results, we decided to not evaluate the UNet-architecture (Figure **??**) but rather work with the other approaches from Section **??**.

In Figure **??** the accuracy results of the different architectures on the test set are illustrated. One can see that the Densenet Combined model performs best, closely followed by the Densenet, Bottleneck and Densenet Combined 2 approach. The Early and Late models perform the worst.

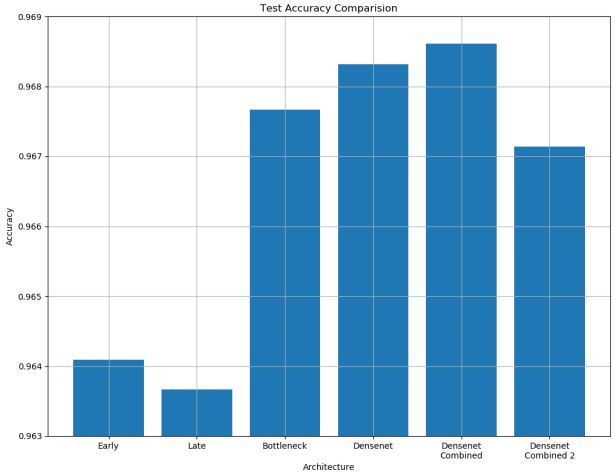
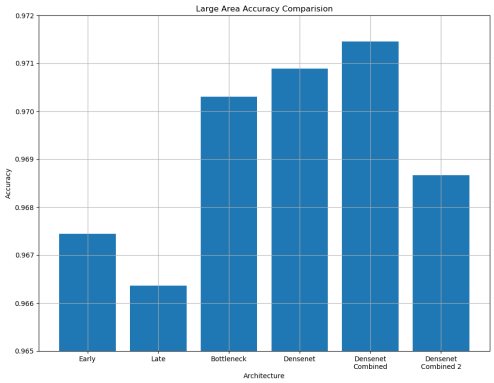


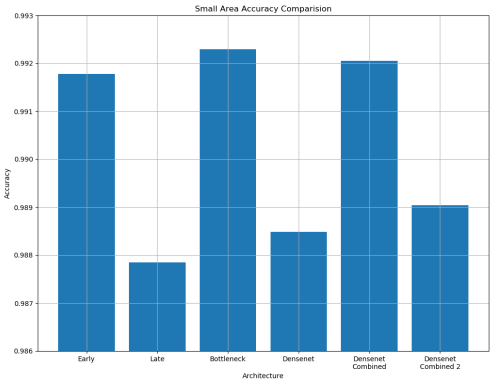
Figure 13: Accuracy for the different image segmentation architectures, evaluated on the test set. For the accuracy higher values mean better results. The best results are achieved with the Densenet Combined approach (Figure **??**).

Figure **??** shows the accuracy of the different architectures on the sub-categorized test sets: large area (Figure **??**), small area (Figure **??**), large flow (Figure **??**) and small flow (Figure **??**).

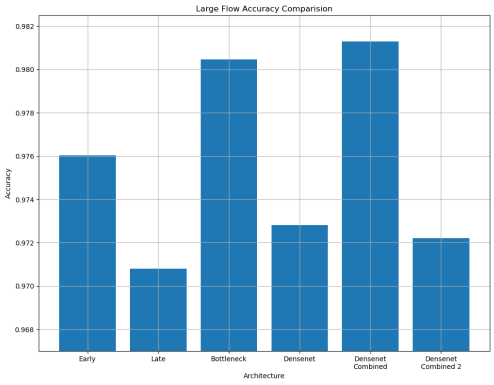
The overall best architecture, Densenet Combined, also performs well on each of the single sub-test sets. The Bottleneck model also performs well on each of these subsets. The overall poor performing Early model achieves great results on the small area test set. Whereas the Late model performs on each single test set poorly.



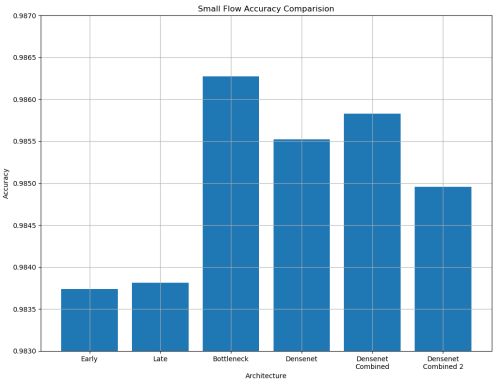
(a) Large area test set



(b) Small area test set



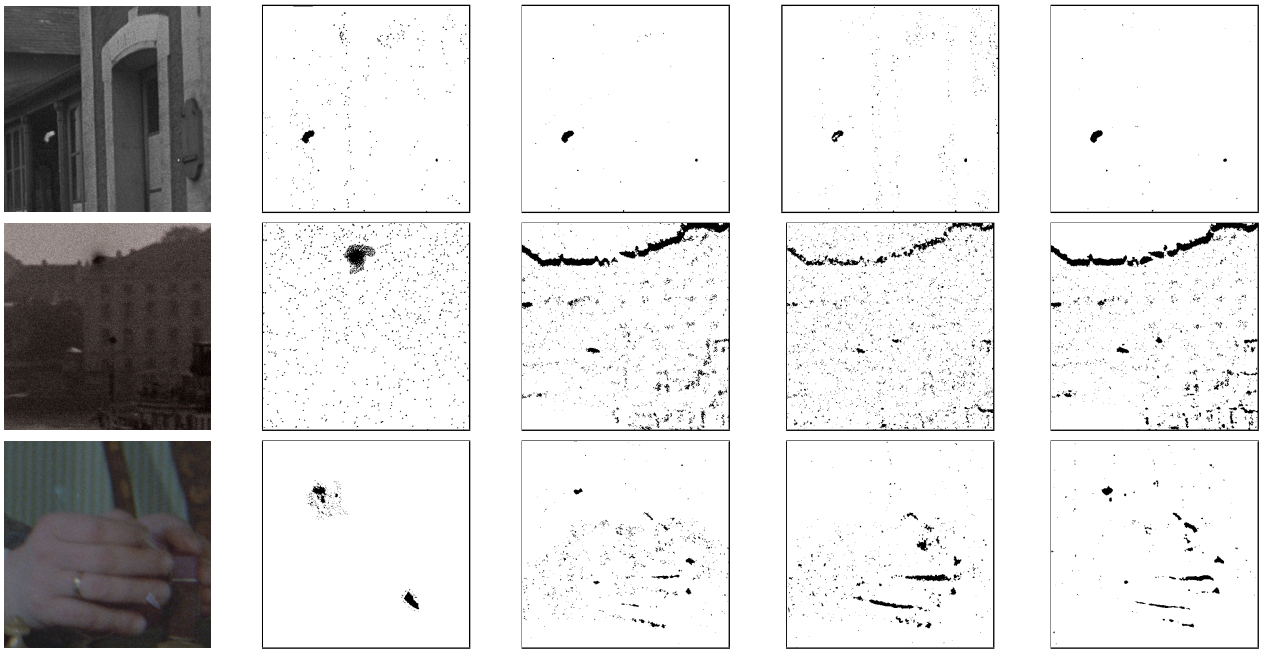
(c) Large flow test set



(d) Small flow test set

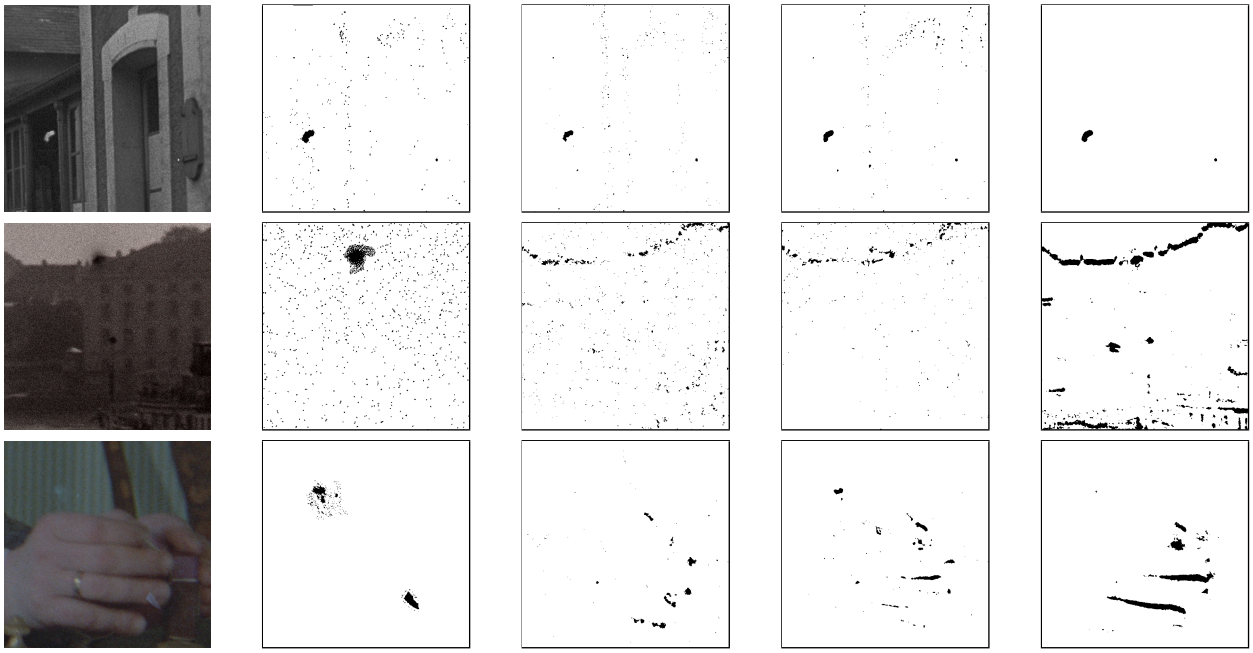
Figure 14: Accuracy for the different image segmentation architectures, evaluated on the sub-categorized test sets.

Figures **??** and **??** give an overview of estimated segmentation masks of the different architectures compared to the ground-truth masks.



(a) Input (b) GT (c) Early (d) Late (e) Densenet

Figure 15: Sample results 1 of image segmentation using additional information for Early (Figure **??**), Late (Figure **??**) and Densenet (Figure **??**).



(a) Input (b) GT (c) Densenet Combined (d) Bottleneck (e) Densenet Combined 2

Figure 16: Sample results 2 of image segmentation using additional information for Densenet Combined (Figure **??**), Bottleneck (Figure **??**) and Densenet Combined 2 (Figure **??**).

The first example in Figure **??** and **??** show the impurity segmentation for a sequence with few motion. One can see that the resulting segmentation masks are quite similar compared to

the ground-truth.

The last two rows in Figure **??** and **??** visualize sample results for sequences with large optical flow. In one example there is a large hand movement, whereas in the other there is a large vertical camera motion. The resulting segmentation masks in these examples is not able to estimate the impurities. The large optical flow has a too large impact into the impurity detection.

5.3.3 Multiple Warped Input Images

As already mentioned we trained all the CNN architectures once for the warped image instances and once without warping to see if there is a difference. The results in Figure **??** suggest that by using warped instances for the previous and next images we can improve the segmentation of the impurities. The best results were achieved with the bottleneck architecture and warped image instances. Interestingly, for all variants of the densenet structure, training with unwarped data achieves better results.

Figure **??** and **??** show results for architectures trained on the warped instances for previous and next images. Figure **??** and **??** show results for the same sequences but with unwarped previous and next images. The first and second example are sequences with low optical flow. For these examples both training methods achieve equal good results. The last example has a large optical flow and the training on the warped image instances achieves better results.

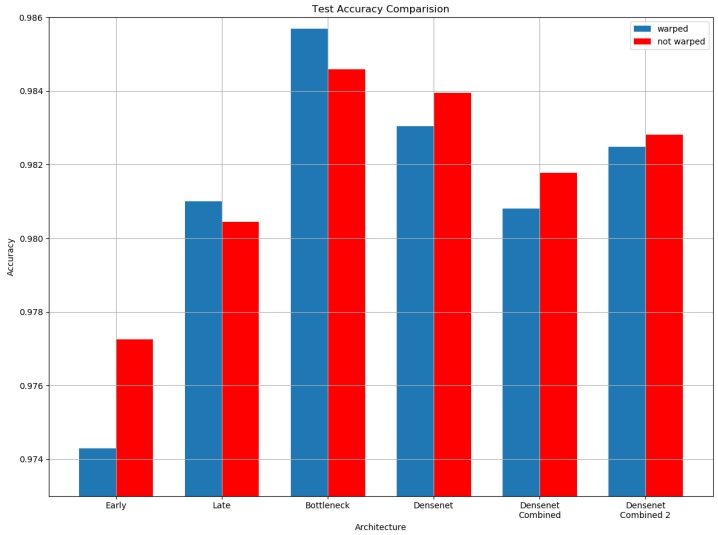
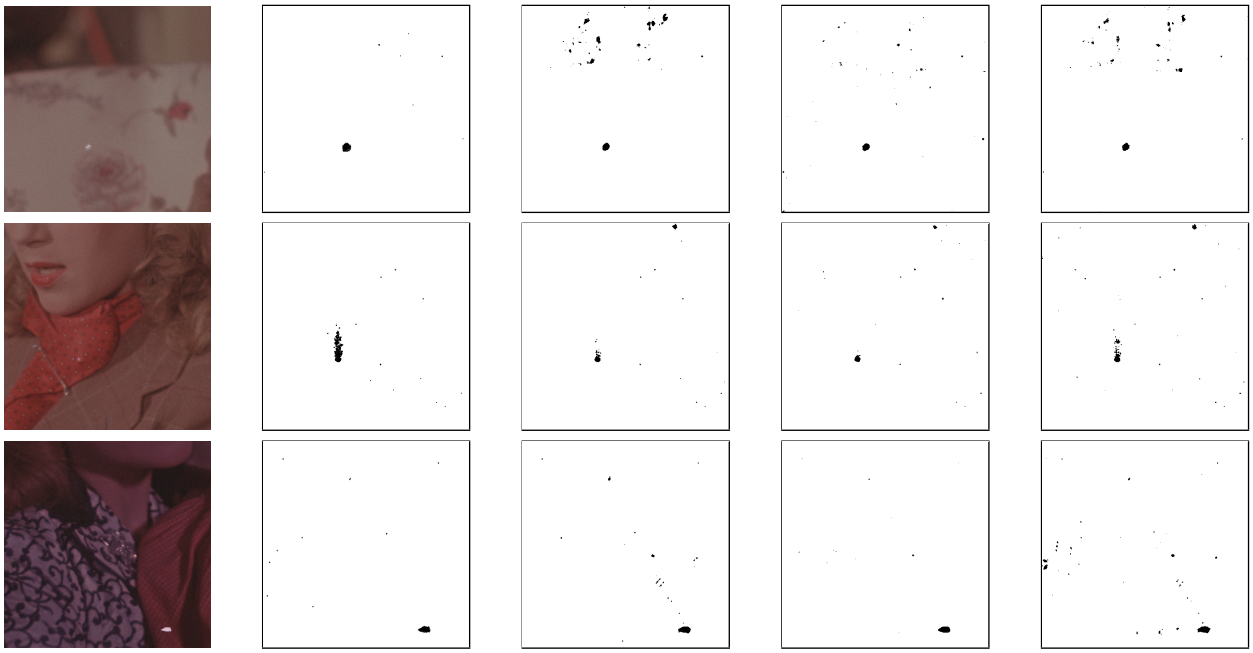
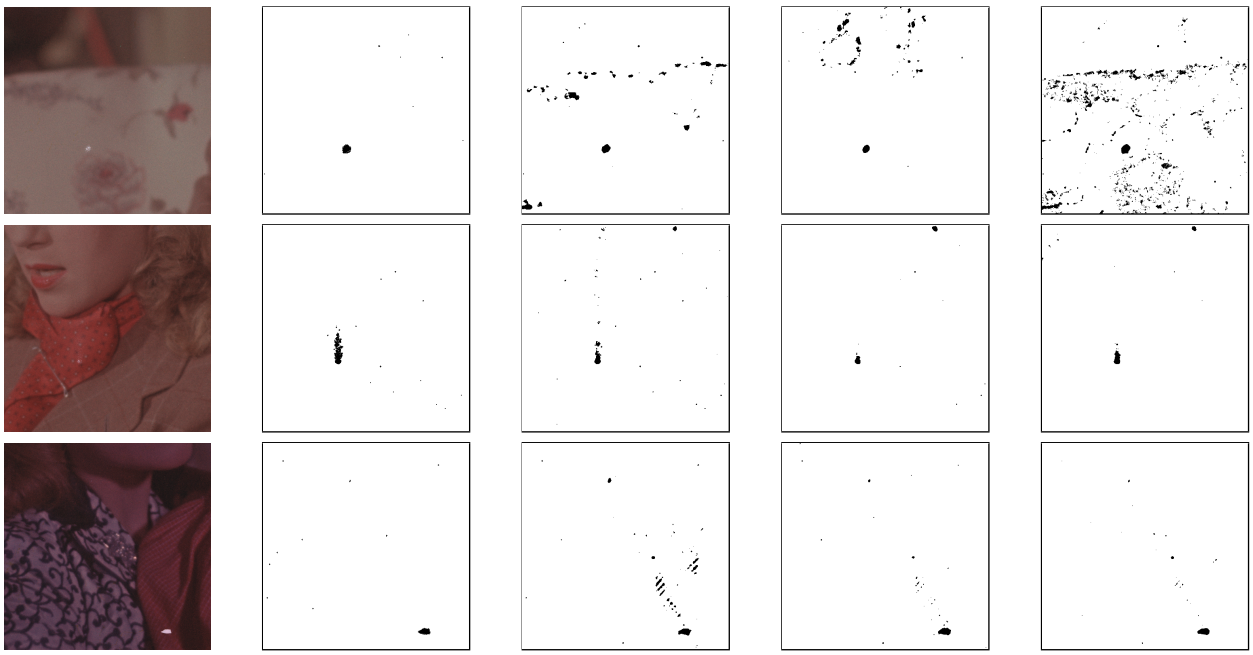


Figure 17: Accuracy for the different image segmentation architectures for warped and unwarped image instances. For the accuracy higher values mean better results. The best results are achieved with the Bottleneck approach (Figure **??**) and warped previous and next images.



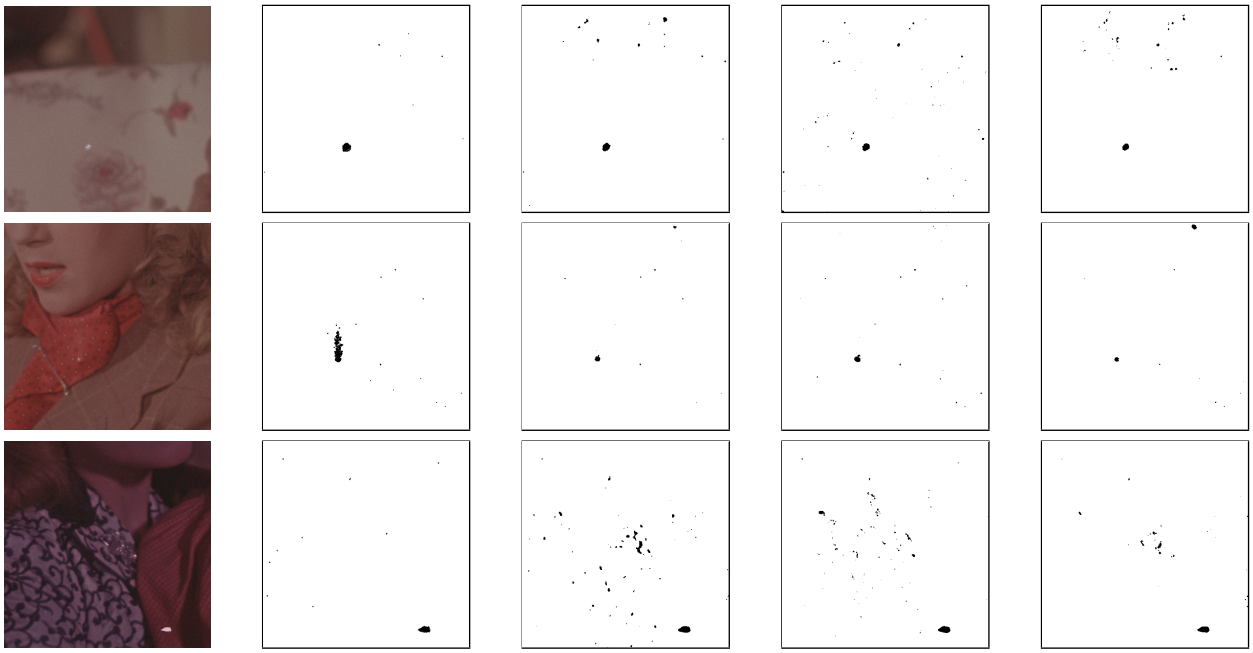
(a) Input (b) GT (c) Early (d) Late (e) Densenet

Figure 18: Sample results 1 of image segmentation using additional warped information for Early (Figure **??**), Late (Figure **??**) and Densenet (Figure **??**).



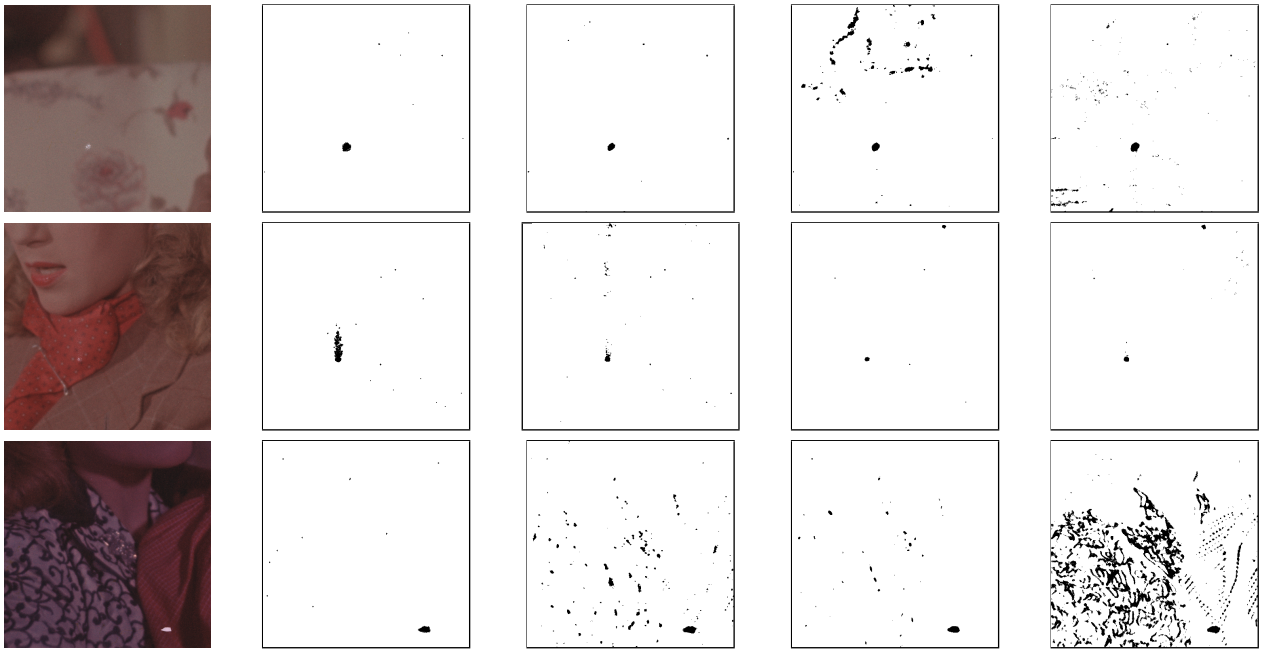
(a) Input (b) GT (c) Densenet Combined (d) Bottleneck (e) Densenet Combined 2

Figure 19: Sample results 2 of image segmentation using additional warped information for Densenet Combined (Figure **??**), Bottleneck (Figure **??**) and Densenet Combined 2 (Figure **??**).



(a) Input (b) GT (c) Early (d) Late (e) Densenet

Figure 20: Sample results 1 of image segmentation using additional unwarped information for Early (Figure **??**), Late (Figure **??**) and Densenet (Figure **??**).

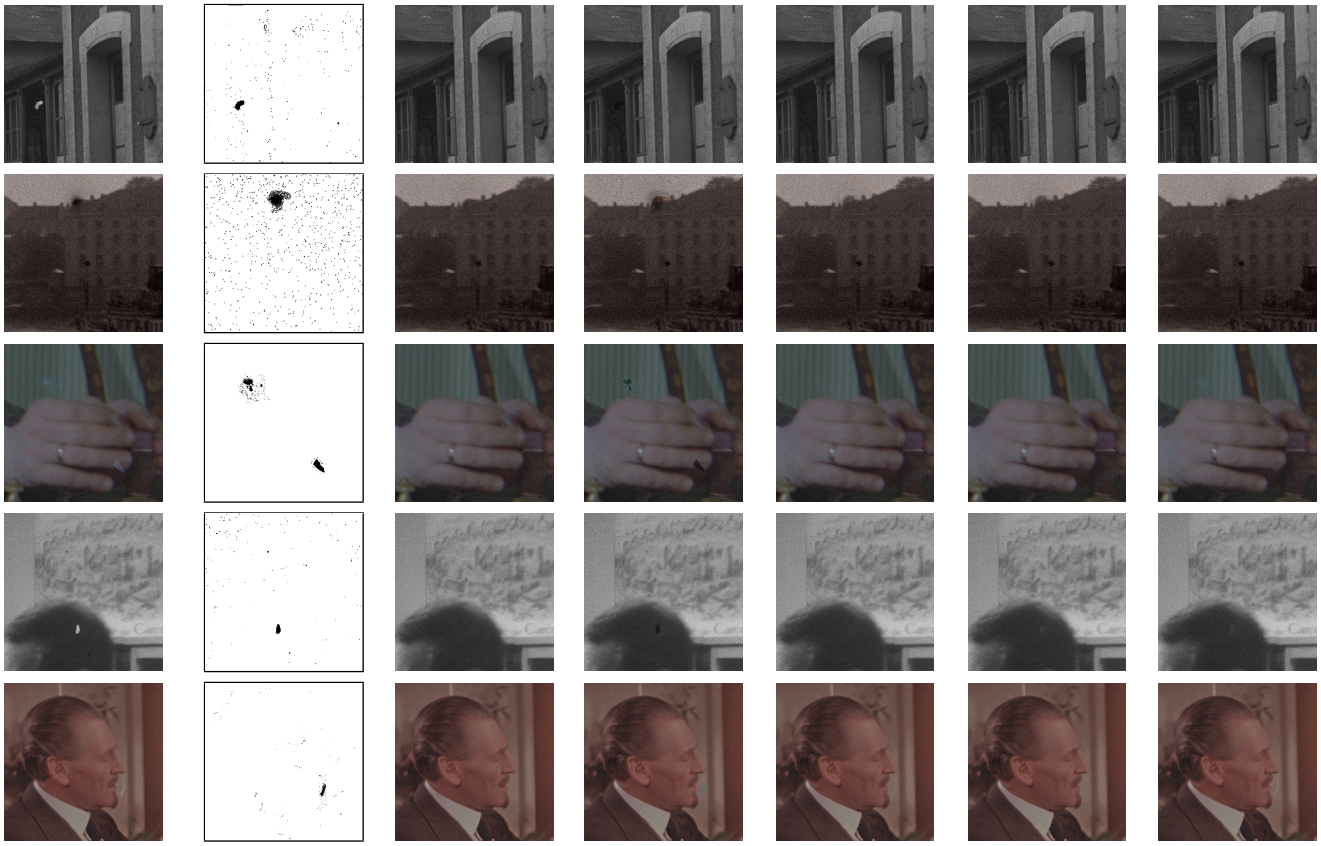


(a) Input (b) GT (c) Densenet Combined (d) Bottleneck (e) Densenet Combined 2

Figure 21: Sample results 2 of image segmentation using additional unwarped information for Densenet Combined (Figure **??**), Bottleneck (Figure **??**) and Densenet Combined 2 (Figure **??**).

5.4 Results for Image In-painting

Figure **??** gives an overview of the results of different image in-painting approaches. One can see that the Densenet approach achieves the visual most appealing image in-painting results followed by the Densenet paths and the Perona-Malik method. The U-Net approach has the visual worst results.



(a) Input (b) Mask (c) GT (d) U-Net (e) Densenet (f) Densenet paths (g) Perona-Malik

Figure 22: Sample results of image in-painting for U-Net (Figure **??**), Densenet (Figure **??**), Densenet paths (Figure **??**) and Perona-Malik (Section **??**).

6 Conclusion

We have introduced a deep learning method to perform image restoration. We showed that a two part approach, splitting into the detection of the impurities and the correction of them, is able to produce visually appealing results.

The impurity detection based on image segmentation as in Section **??** works best with the Densenet Combined model (Figure **??**). By investigating segmentation mask results (Figure **??** and **??**) one can see that for sequences with high flow the resulting segmentation mask is quite wrong compared to the ground-truth mask. The approach to use warped instances of the previous and next image (Section **??**) to overcome this behavior, suggests that the Bottleneck model (Figure **??**) produces even better results. The approaches from Section **??** and **??** could better be compared if we have warped images instances for all sequences.

Also the segmentation using just a single input image does not work at all. This may be due to the fact, that the impurities often are not that distinctive compared to the overall structure of the images. Adding additional information from previous and next images improve the results, since most of the time the impurities on the current frame are not visible in the previous and next image (Figure **??**) and the CNNs are able to suppress the structural properties of the images. The approach with unwarped image instances works bad for sequences with high optical flow. This may be because that pixels concealed in one image but visible in another are treated just like impurities.

The correction of the impurities based on image in-painting works visually best with the Densenet model (Figure **??**). Other approaches, like *e.g.* an encoder-decoder scheme or an approach based on GANs may achieve better results, but they may be more complex to train. The Perona-Malik approach may be improved by using an adaptive variant of the coherence enhancing diffusion, like in the work of Prasath [**?**], but as a relative efficient model, it works quite well.

In general a higher number of training and test data would improve the results of our CNN approaches. Supervised deep learning approaches need a huge amount of training data to perform well. Also, some of the provided ground-truth masks are of poor quality, which makes it hard for the networks to learn sufficiently.

The proposed method can be adapted to a wider range of applications. By using different training data, it can be used, *e.g.* to remove objects in images or for the removal of overlays.