UNIVERSITEIT VAN AMSTERDAM

BACHELOR THESIS; CREDITS: 18 EC

Frame Interpolation using Convolutional Neural Networks on 2D animation

Author:
Haitam BEN YAHIA
StudentID:10552359

Bachelor Opleiding
Kunstmatige Intelligentie
University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor:
Matthias REISSER

PhD Researcher
QUVA Lab
Room C3.249
University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Used software:
Linux Ubuntu 14.04, Windows 7, Mac OSX 10.10, Python 2.7.10 & TensorFlow 0.8

August 5, 2016

Abstract

Frame interpolation is a computer vision task that is largely performed on real life video to increase the number of frames. 2D animation could also benefit from this as drawing each individual frame takes a considerable amount of time. Existing 2D animations can also benefit from this by increasing the current frame rate and by increasing the number of frames. Proper frame interpolation requires an understanding of the subjects within frames to accurately perform. Convolutional Neural Networks (CNNs) have been proven to be very good at object detection and could aid the process of specific frame interpolation. Frame interpolation has been done before using CNNs on real-world videos. 2D animation differs from real life video in important ways. Several approaches using CNNs have been proposed in this thesis to perform frame interpolation in the domain of 2D animation. Results show that these methods do not perform as well on 2D animation as it does on real life video, because of larger object displacements, lack of constant motion and overall noise caused by stylistic choices.

Contents

1 Introduction

2D animation consists of hand drawn frames that together form a motion sequence. These drawings try to capture many principles that are present in real life objects which move according to laws of physics. They simulate the same consistencies between images from real life video, such as shape, color, contrast, objects and lighting. 2D animation also introduces other aspects that have to be consistent to simulate real life motion such as design and style. Drawing these frames can require a lot of manual labor because of the consistency and precision that is needed.

The convention in animation is to use at least 24 frames per second (FPS) to create smooth transitions between drawings. Though the same drawings could be repeated when little to no movement is present. Thus, the amount of drawings within a second can vary from one to 24 FPS in most cases. These repeated drawings come at the cost of misrepresenting real life video motion, where motion is generally present in similar cases. An example would be a video of a person in an idle position. 2D animation would just show a drawing of the person whereas real life video would capture subtle movements such as hair moving as a result of the wind blowing outside. Ideally, this would also be animated to enhance the sense of characters being alive, however these subtle movements are disregarded when there are no animators available or there is no budget available to hire them.

Animation makes the distinction between key frames and so-called in-between frames. These key frames indicate the most prominent poses of a movement. The in-between frames function as transitioning frames from key-pose to key-pose. These in-between frames are thus characterized as not having many sudden deviating object positions, compared to their neighboring frames. Furthermore, they make up most of the frames present in animation. These frames could potentially be generated considering the small displacement of objects. Auto generated in-between frames have the advantage of not requiring manual labor. It can also be used to increase the current number of drawings to make a particular motion appear slower. Lastly, it can replace the duplicated frames for a smoother motion. Generating intermediate frames between existing ones is called also frame interpolation.

This thesis researches the question of whether it is possible to use frame interpolation in the domain of 2D animation. This, in particular, is done with convolutional neural networks (CNNs) as these have shown to be versatile in computer vision tasks shown by Meyer et al.[6] Long et al.[5] and He et al[3]. Additionally, these networks have been improved to the point where it could be rivaling human level vision at object recognition according to He et al[3]. To make this question simpler, however, no distinction is made between key frames and in-between frames because data of sufficient size and with frames labeled as such is unavailable. Data is required for a CNN to train and learn from and makes it an essential part of the algorithm. Instead an attempt has been made to generate the middle frame of three consecutive frames derived from an arbitrary 2D animated sequence. Only the middle frame has been chosen to be recreated using frame interpolation considering the large displacement of objects that occur between larger amounts of intermediate frames in larger sequences. See Figure 1 for an example of three consecutive drawings and an overlay of frame 1 and 3. All 3 frames are different, but contain little motion in this example. Notice the right arm and the hair in the overlay.

Figure 1: From left to right: ground truth (frame 2), frame 1, overlay of frame 1 and frame 3, and frame 3.

2 Literature Review

Frame interpolation, a subject of computer vision, is traditionally done with algorithms that do not learn from large amounts of data, but rely solely on the input image sequence to achieve this task. The current state of the art algorithm attempts to solve this by using *optical flow estimation*[1] according to Meyer et al[6]. This algorithm matches the same pixels within a sequence of images after which frame interpolation is achieved by averaging a pixel between a pixel pair. He et al.[3] state that CNNs have led to a series of breakthroughs for image classification and can recognize low, mid and high-level features such as edges, faces and entire objects respectively. Studies by Long et al.[5], Fisher et al.[1] and Teney et al.[7] show that optical flow estimation is possible using a CNN . There is little to no research done on frame interpolation while using CNNs as a focus. However since the task of optical flow estimation is closely related to frame interpolation, many aspects could proof to be insightful or could be used as building blocks for a frame interpolation algorithm using CNNs.

Frame interpolation within the domain of 2D animation differs from frame interpolation in real world video in several ways. Motion in 2D animation is less consistent compared to real life motion because it is not bound by fundamental laws of physics. 2D animation conventions also hold a particular way of producing so-called in-between frames, which is what this thesis attempts to generate accurately. Moreover, the low amount of frames per second generally causes larger displacements between objects within an image sequence. This, in particular, is problematic for interpolation algorithms according to Meyer et al[6]. Classifying objects could prove useful with ambiguous motions being possible in animation due to its potentially large displacements. CNNs automatically choose their own features and certain object recognition features could be present when doing specific frame interpolation, which might lead to specific frame interpolation for specific objects. Long et al.[5] showed that optical flow estimation can be learned by feeding large amounts of video data with frame interpolation as a byproduct. Their focus was not frame interpolation and thus their research consisted mostly of improving optical flow estimation.

Long et al.'s[5] architecture consists of several convolutional layers followed by several deconvolutional layers. This architecture can also be seen as an encoding/decoding structure. It transforms an image pair into a tensor many times smaller, which in turn is used to reconstruct an interpolated frame. A study by He et al.[3] shows that residual connections in the CNN architecture improve classification when increasing the amount of layers. Residual connections are adopted in a different way by Long et al.[5], to ensure that lost information through the convolutional part is reintroduced in the deconvolutional part. Fisher et al.[1] used a similar architecture as Long et al.[5] for optical flow estimation, but showed that convolving the input images separately and combining the weights at a later stage improved optical flow estimation on certain data sets. Since optical flow estimation is a subproblem of frame interpolation, Fisher et al.'s[1] architectures could be used as building blocks for a better frame interpolation architecture and is proposed in Section 3.2.4 as the double input

[1] Optical flow estimation is the task of estimating relative motion between the observer and the scene in terms of edges, objects or surfaces

model.

3 Method

3.1 Data

3.1.1 KITTI RAW

There are two data sets that are used in this thesis beginning with the KITTI RAW image data set, which consists of images from various residential [2] and city areas[3] captured using a camera attached to a driving car and is made by Geiger et al[2]. KITTI RAW is used to compare results to Long et al.[5], who also use this, with the initial model proposed in Section 3.2. It consists of 18958 images from 37 scenes or 18884 triplets. All the frames have been scaled down to 64×196 pixels to reduce computational time. The frames are read in PNG format and are decoded to raw 8bit RGB-values containing 255 possible values for each channel. Triplets have been made by taking three consecutive frames for every frame except the last two of each scene because these lack following frames.

3.1.2 Animation

The second data set consists of images derived from various animated TV shows. These images have been extracted from 2 YouTube videos, which are comprised of specific scenes where a lot of motion is present. The first video[4] consists of scenes where there is rotation of a subject or rotation of the camera/observer present. These are generally easier to draw in-betweens for because of smaller displacement of objects. The second video[5], which is roughly 10 times as long, is a more general case with animation. These contain scenes that are not limited to a particular motion like camera footage from a driving car and is sourced from various animated TV shows in the month of April 2016. Both videos have been converted and stretched down to images of 160×96 pixels. This number has been chosen to closely resemble the original aspect ratio of the YouTube video (16:9), but also to make it compatible with the CNN models proposed in Section 3.2. The model requires the image size to evenly be divided by 2 at least 5 times. The animation frames are read in PNG format and are decoded to raw 8bit RGB-values containing 255 possible values for each channel as well. Triplets have initially been made the same way as in the KITTI RAW data set. The data set consists of 32526 images. Some scenes have been extracted to be used for qualitative evaluation in Section 4.2, which have not been trained or tested on.

3.1.3 Pre-processing Animation Triplets

The animation triplets initially contain a lot of entries that do not help learn the model to do frame interpolation and might even hinder this. These will be referred to as invalid entries and can be considered as noise. These include triplets consisting of the same images and do not contain motion, as well as triplets that are in between two scenes and thus break a sequence, and finally triplets that are ambiguous because they consist of objects that are displaced too far apart. See Figure 2 where frame 2 could not be generated given only frame 1 and 3. This is because the object could be moving in several different directions to fit in this sequence. The KITTI RAW data set does not contain invalid triplets because of relatively small object displacements and constant motion. These invalid triplets are filtered out by comparing absolute values of the images. This is done by computing the difference between two frames as $D(f, \bar{f}) = \sum_{i=1}^{96} \sum_{j=1}^{160} \sum_{k=1}^{3} |f_{ijk} - \bar{f}_{ijk}|$, where D is the difference, and f and $\bar{f}$ are the frames that are compared to each other. All triplets that do not satisfy

[2] http://www.cvlibs.net/datasets/kitti/raw_data.php?type=residential as of June 2016
[3] http://www.cvlibs.net/datasets/kitti/raw_data.php?type=city as of June 2016
[4] https://www.youtube.com/watch?v=haBdcjEHUuc as of June 2016
[5] https://www.youtube.com/watch?v=-zM4dKiF0-I as of June 2016

$T(0.01) < D(frame1, frame2) < T(0.22) \wedge T(0.01) < D(frame1, frame3) < T(0.22)$

are discarded, where $T(t) = 96$ (px)$\times 160$ (px)$\times 3 \times 256$ (RGB-value)$\times t$. The values 0.01 and 0.22 have been set according to the animation data set and has been evaluated by manually going over a small portion of the triplets that were classified as invalid. Valid triplets are then compressed in a file which is loaded when training starts. 23047 valid triplets remain after pre-processing.

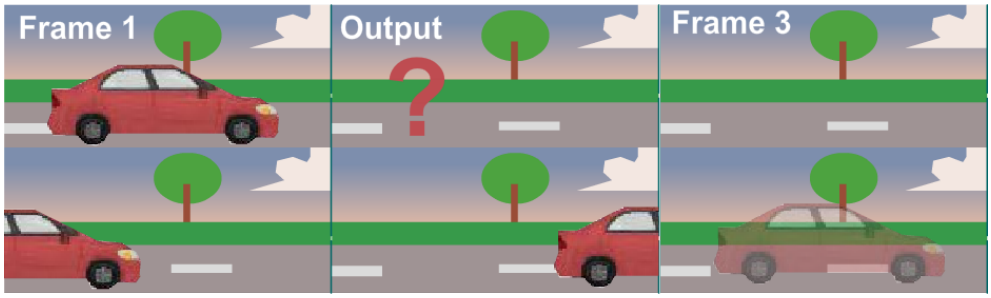

Figure 2: The top row shows the first and 3rd frame, whereas the bottom row shows possible 2nd frames that would fit in the motion

3.2 Model

The model of the CNN in this thesis is largely inspired by Long et al.[5] who use so-called convolutional and deconvolutional blocks . These blocks consist of convolutional layers with a 3×3 filter (CONV), Rectified Linear Units (ReLU), and a max pooling Layer with a 2×2 filter (POOL) or a deconvolutional layer with a 4×4 filter (DECONV), which are elaborated on in Section 3.2.1 and 3.2.2. The model in this thesis adopts the convolutional block as the following structure:

$$INPUT -> [CONV -> ReLU] * 3 -> POOL$$

The decovolutional block is also adopted in the structure:

$$INPUT -> DECONV -> ReLU -> [CONV -> ReLU] * 2$$

See Figure 3 for a graphical representation of the blocks. The blocks differ from the Long et al.[5] model in that it uses ReLUs instead of Parametric RelUs. The reason for this is because parametric ReLUs are currently not implemented in TensorFlow[6], which is used to create these models. TensorFlow is a software library for numerical computation using data flow graphs and eases the process of implementing machine learning algorithms within a shorter time period. See Figure 4 for a graphical representation of the model. This model will be referred to as the initial model hereafter. Notice the input is two images which are concatenated on the RGB-channel's dimension.

[6]See more information on TensorFlow at https://www.tensorflow.org/

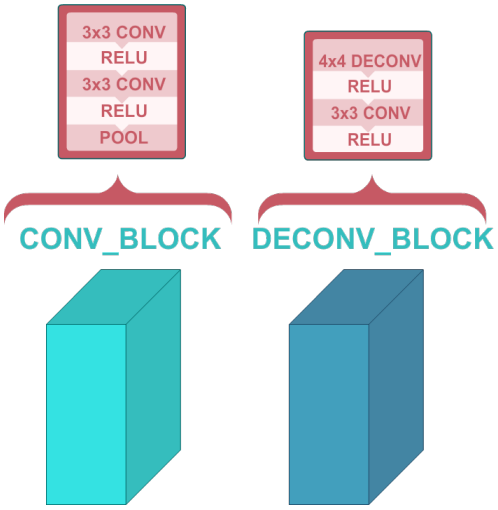

Figure 3: A graphical representation of convolutional and deconvolutional blocks which is used in Figure 4

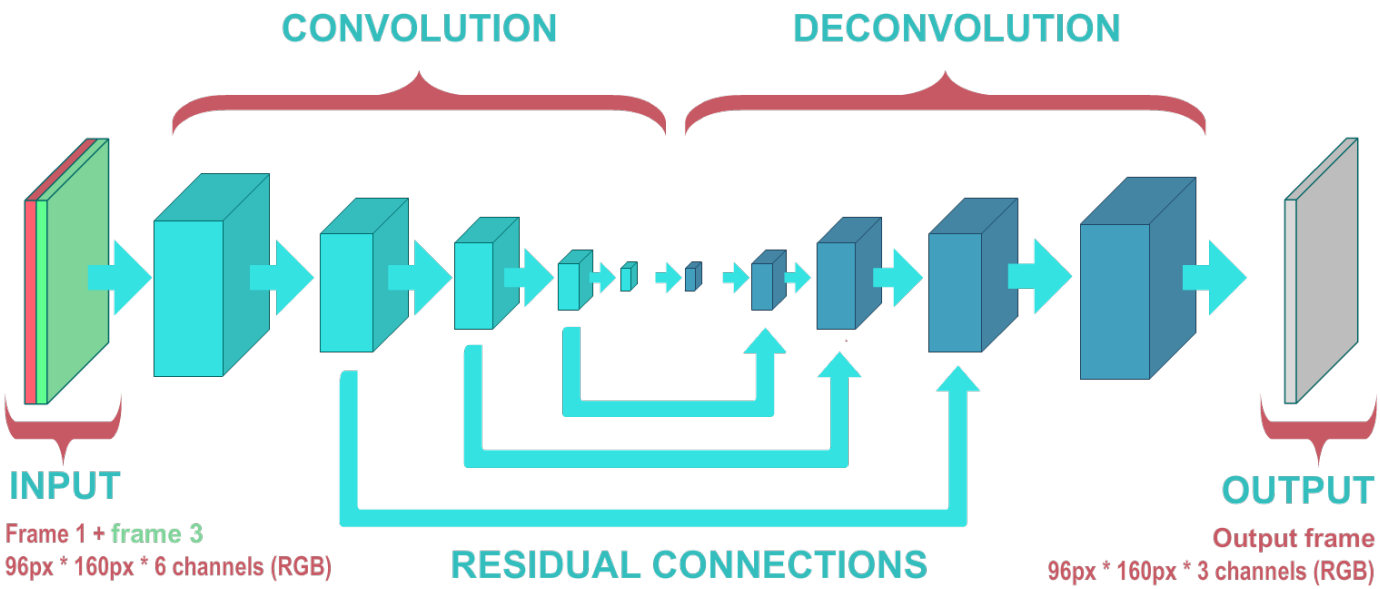

Figure 4: A graphical representation of the initial model

3.2.1 Convolutional block

The convolutional layers are initiated with random weights and the bias is initiated with slightly positive numbers. Given the activation maps produced by convolutional layers, RelUs will threshold the maps at 0. Every convolutional block ends with a 2×2 max pooling layer resulting in an output that reduces the height and width to half its original size. This means that a $96\text{px}\times160\text{px}$ input will be reduced to a 3×5 feature space. Deconvolutional layers are used to reconstruct this feature space back to an image of its original size. See Table 1 for size information on each block used in all models proposed in this thesis.

3.2.2 Deconvolutional block

The deconvolutional block is very similarly structured to the convolutional block. The only difference is that the first convolutional layer is replaced by a deconvolutional layer and the

max pooling layer is removed. The deconvolutional layer tries to reconstruct the input on a larger scale. It does this traditionally by taking the transpose of the filters and the output of a convolution, after which the input will be reconstructed. The output size of this layer can be specified with the output stride, which ables the network to eventually create an output of greater size. The filters, in this case, will be trained to reconstruct the input given less information, because there is information loss when performing a convolution this up-sampling will not be able to perfectly reconstruct the input. See Table 1 for size information on each block.

	Input	CONV1	CONV2	CONV3	CONV4	CONV5	DECONV5	DECONV4	DECONV3	DECONV2	DECONV1	Output
height	96	48	24	12	6	3	6	12	24	48	96	96
width	160	80	40	20	10	5	10	20	40	80	160	160
depth	6	96	96	128	128	128	128	128	128	96	96	3

Table 1: The size of each convolutional and deconvolutional block

3.2.3 Residual connections

Because the max pooling operation reduces the feature space to $3\times5\times128$, the model loses a large amount of fine-grained details. The feature space is scaled down for at least two important reasons. Firstly, the complexity of the computation goes down significantly. Secondly, the deepest layers forces the model to learn high-level abstract features and also the position of these features in the original image. These are essential because of the high-level feature differences between input images, such as a moving face, would be captured here as well. Thus models by Long et al.[5] and Fisher et al.[1] introduced residual connections to counter this information loss. The activation maps produced by the primary convolutional blocks retain fine-grained information such as background detail that is coherent in both input images. These activation maps are concatenated to the activation maps produced by the deconvolutional block and are then fed as the input for the next deconvolutional block. Required from both types of activation maps are to have the same width and height. The number of activation maps fed into the next deconvolution block is equal to the sum of maps derived from the output of the previous block and the block from the residual connection. This is illustrated with the light blue arrows in Figure 4 and 5.

3.2.4 Double Input Model

Modifications were made to the initial model to potentially improve frame interpolation performance. This proposed double input model is inspired by *FlowNetC* from Fisher et al.[1] as they suggested that this model might improve optical flow estimation given a different data set. It is not specified what differences could improve this, but the animation data set differs in some essential parts from real life video described in Section 1. These differences could have had a positive effect on the proposed model. The convolutional blocks in this modification stay the same, the only difference is that the first two convolutional blocks each convolve one image as apposed to two images stacked together. See a graphical representation in Figure 5 where this is illustrated. The outputs of both second blocks are concatenated after which they are sent to the third convolutional block and is also concatenated with the output of the second deconvolutional block as a residual connection.

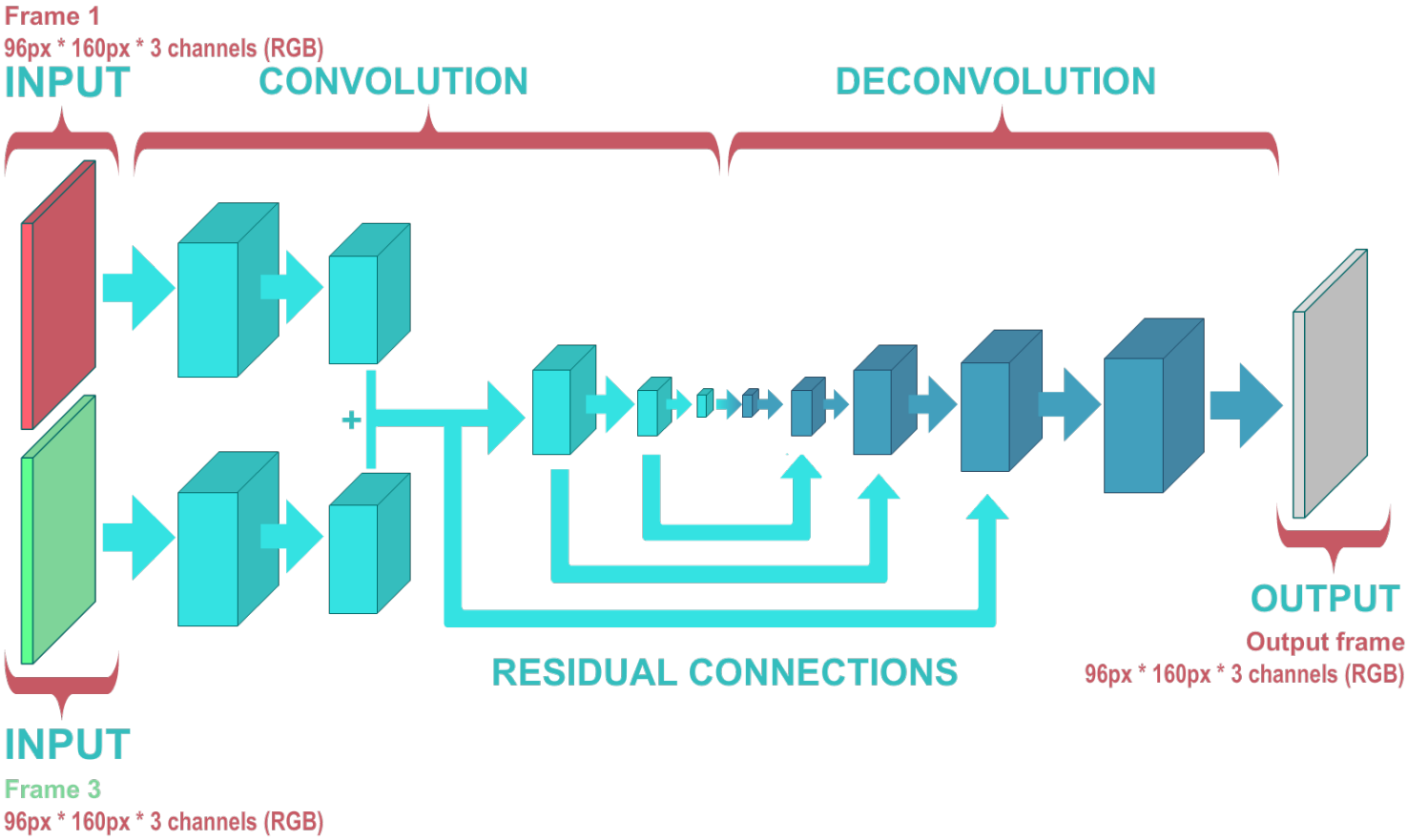

Figure 5: graphical representation of the double input model

3.3 Training details

There are several training details that are of importance to recreate the same results. The triplets have been augmented by flipping the images vertically and are thus up-sampled for KITTI RAW and the animation data set to 37824 and 44690 triplets respectively. This has been split to 75% of the data as the training set and 25% as the test set. The triplets from the same scene are not present in both sets at the same time to reduce overfitting on a particular scene. The training set was shuffled each epoch after splitting the data set. The CNNs are trained on a Nvidia Titan X with a batch size of 8. This number has been chosen to have the maximum batch size the GPU can handle while concurrently running different processes irrelevant to this thesis. To stay consistent with newer result this size has not been altered at a later stage. The learning rate has been set at 0.001 with an exponential decay applied after 8 epochs. This is due to the fact that exponential decay showed slower converging whereas it converges after roughly 8 epochs without it. The learning rate was multiplied by 0.95 every 5000 batch steps with the exponential decay. The weights of the network are optimized with the Adam Optimizer by Kingma et al[4] and is proposed by Long et al[5].

3.3.1 Loss function

The L1 loss function was used, which was minimized during training. This calculates the absolute differences (S) between the ground truth(y) and the estimated values(f) in the output as $S = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{3} |y_{ijk} - f_{ijk}|$. It should be noted that this is not the optimal fit for animation frame interpolation problems. 2D animation contains mostly scenes where the background is static and the objects are moving. The background can easily be reconstructed and learned by the CNN if their pixel values are similar in both input images. The aim

of frame interpolation, however, is to interpolate objects that move and are thus not static. Larger objects are punished more severely for improper frame interpolation as they cover a larger volume of pixels. This means that frame interpolation on larger objects could have performed better by accurately placing objects at the right spot, while the blurriness could cause the same error as in an image with a small object and improper frame interpolation. See Figure 6 where both results produce the same L1 error. Notice the top row which retains the circular shape relatively well considering the large displacement seen in the overlay. The bottom row output shows two round shapes as apposed to one which is identified as improper frame interpolation. The bottom row however contains a volume of pixels that cause roughly the same error as the volume of pixels that cause the error in the top row. This problem is less relevant in video where roughly every pixel differs from the values of the same pixel in their neighbouring frames.

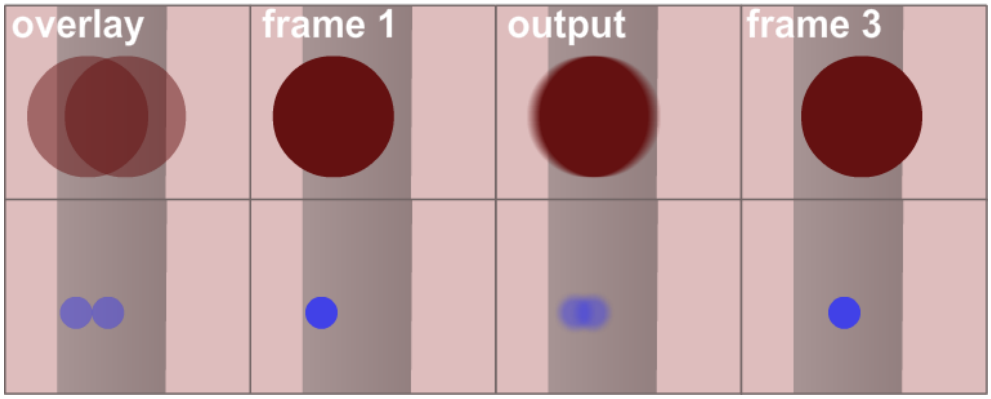

Figure 6: An example of two triplets with roughly the same error with the top row showing better frame interpolation than the bottom row

3.3.2 Difference between input images

A different approach to making the initial model focus on the changes of the input images as opposed to accurately constructing the background has been attempted. By reducing values that are the same in both images to 0 in the second input, the model will adjust the weights and create less filters that concern themselves with the static part of both images. Instead of feeding the network two input images (first and third frame), only the first frame and the area in which the third frame differs from the first frame is fed instead. See Figure 7 to see an example. This is done by comparing the sum for every pixel over the RGB channels of both frames. All pixels summed over the RGB-values in an input frame are subtracted from the same pixels in the second input frame. This can be written as $g_{ij} = |\sum_{k=1}^{3} f_{ijk} - \sum_{k=1}^{3} \bar{f}_{ijk}|$ where the absolute differences are calculated. Afterwards, a mask was made which contained 0 and 1 values for each element in g_{ij} where values are below or above 10 respectively. Compression techniques in video introduce some noise and thus a small error range of 10 has been set to account for this. As a comparison, the maximum difference possible in this equation is $256 \times 3 = 768$ which corresponds to the difference between a black and a white pixel. The mask is applied on all 3 channels of every third frame in a triplet, which results in an image that has values of the third frame where pixels differ from the first frame and zeros everywhere else.

Figure 7: An example of a triplet sequence: ground truth, frame 1, CNN output and the value of ever pixel in frame 3 that differs from frame 1

3.4 Reconstructed Animations

Animations have been recreated by reconstructing every other frame of the original sequence. No distinction has been made here between valid or invalid entries as described in section 3.1.3. This will result in some frames performing less than average as it will still produce an output for these possibly ambiguous triplets. A post-processing step has been applied to retain more fine-grained details in static parts of the sequence. The same pixels within the two input images that share the same value (within a difference margin of 10) will be copied over to the output frame. This is mainly done to regain the sharpness of static objects. Flickering of the backgrounds in these cases is reduced and forces the viewer to disregard reconstruction of static parts in the sequence as part of their evaluation. This has also been done for the individual images shown later in Section 4.2 for the same reason. Frame interpolation will be forced to be the focus in the qualitative evaluation this way and is not distracted from reconstruction of static objects in the input images.

4 Results and evaluation

4.1 Loss curve

The initial model was tested on the KITTI RAW data set to compare results from Long et al.[5] where proper frame interpolation was noticeable. Figure 8 shows the decrease of the loss function over 120.000 batch steps. Notice the converging error to be around 7000. For a sense of scale: the maximum error rate is the absolute difference between a black and white image , which is an array with zeros and an array with 255 as values respectively. The maximum error is thus $64 \times 192 \times 255 \times 3 = 9400320$ between two raw 8-bit RGB-images which leads to a relative error of $\frac{7000}{9400320} = 7.446 * 10^{-4}$ with an absolute error of 7000 in the graph. No overfitting has been observed similarly to findings in Long et al[5].

Comparing this to the results of the initial model using the animation data set, shown in Figure 9, shows some key differences. The error range between each step varies more in the animation set because this data set contains significantly more scenes that differ from each other in terms of color, subject, motion and overall content. The loss averages at an L1 error around 12000, which is much higher than on the KITTI data set. However, it should be noted that these images are of different size and allow for larger errors, which also contributes to a larger error range. The maximum error is $96 \times 160 \times 255 \times 3 = 11750400$ which leads to a relative error of $\frac{12000}{11750400} = 1.021 * 10^{-3}$ with an absolute error of 12000. This relative error is done to compare error results between the KITTI RAW data set and the animation data set which have different dimensions. This is still a higher error than the KITTI RAW error at convergence.

The loss curve of the double input model in Figure 10 shows relatively no difference when compared to the initial model. This is despite the results in Fisher et al.[1] which showed slight overfitting with their data set. The loss curve of the different input approach discussed in Section 3.3.2 is shown in Figure 11. It starts off similarly with an average L1 error around 12000 but overfits rather quickly on the training set.

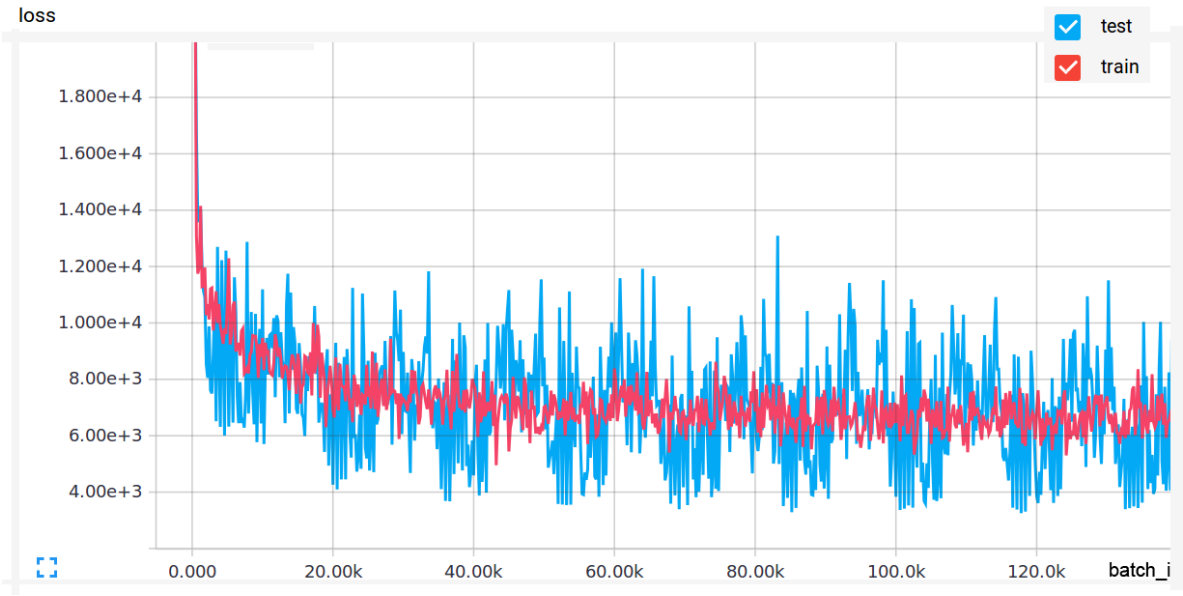

Figure 8: KITTI RAW loss curve on initial model with frame1 and frame 3 as input within every triplet

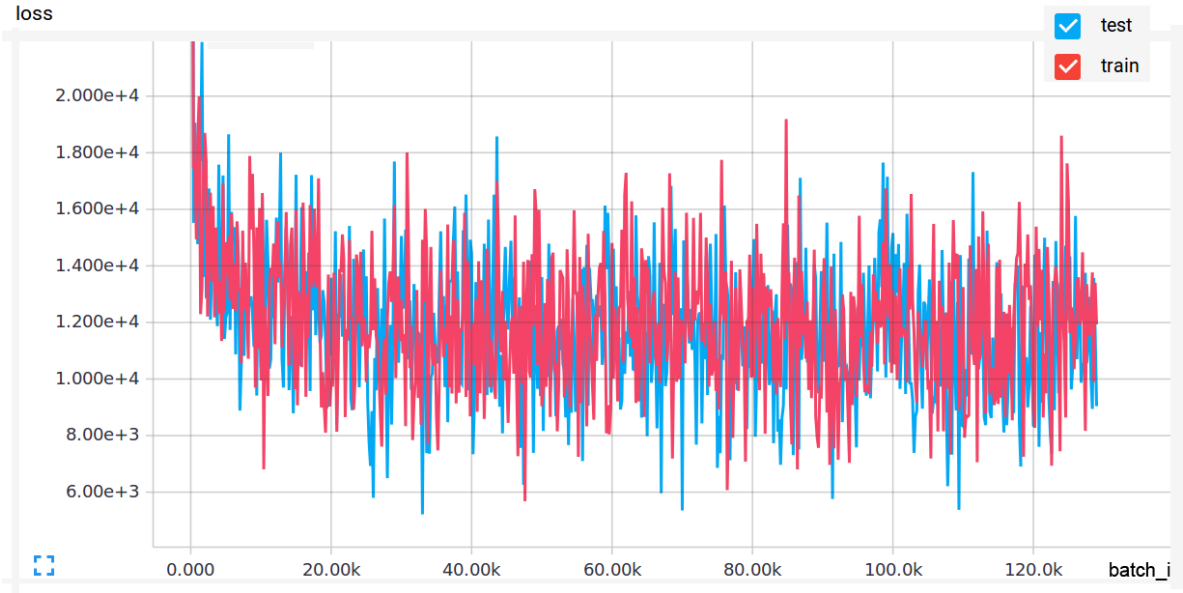

Figure 9: Animation data set loss curve on initial model with frame1 and frame 3 as input within every triplet

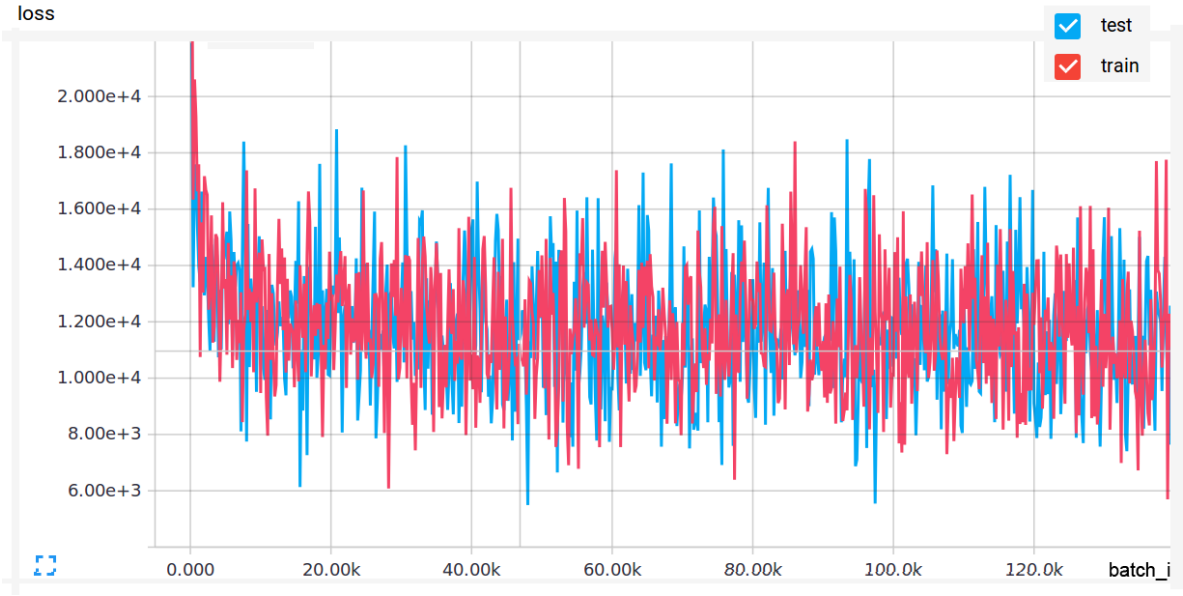

Figure 10: Animation data set loss curve on the double input model with frame1 and frame 3 as input within every triplet

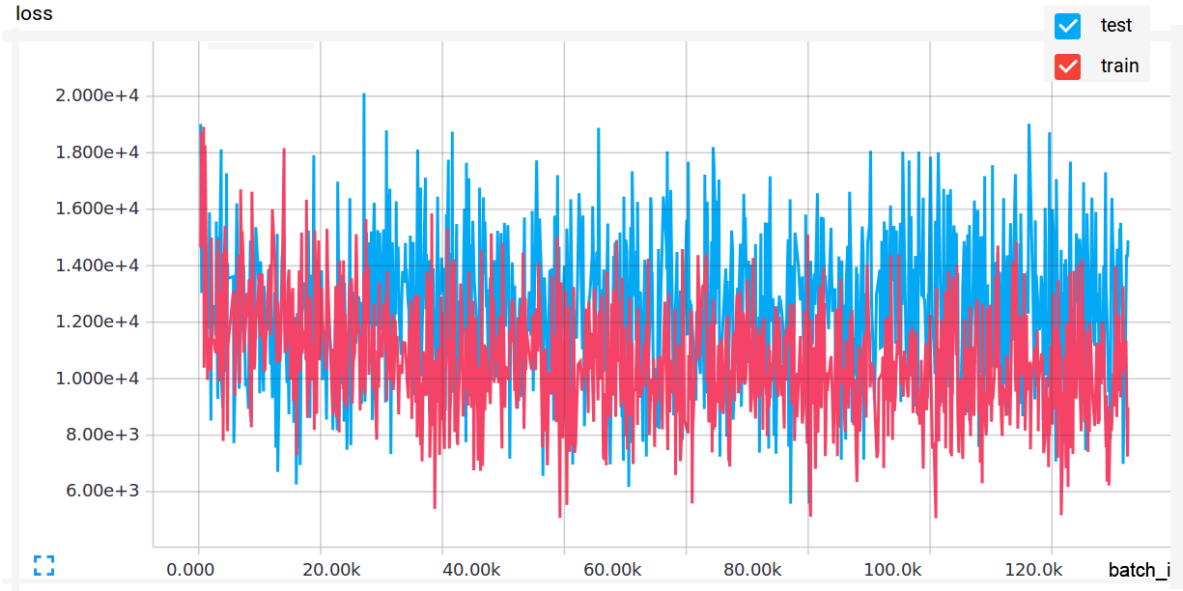

Figure 11: Animation data set loss curve on initial model with frame1, and the difference between frame 1 and frame 3 as input within every triplet

4.2 Triplets

A qualitative analysis is needed, because the error is not a relative measure for how well it does frame interpolation in 2D animation. The first set of analysis confirmed whether results are similar to Long et al.[5] done with the KITTI RAW data set. Figure 12 shows an example of a triplet in the validation set. Notice the shadow in the lower left corner which interpolates well. Some blurriness is present and it is not perfectly sharp, however edges blobs and overall shapes are in the right place. Notice the small patch of the wall surrounded by wall vines near the right edge in frame 1, which is not present in frame 3. In the output and the ground truth,

the patch is placed against the right edge of the frame. Frame interpolation has been spotted in all triplets, even though some are more blurry than others.

Figure 12: KITTI RAW example triplet, from left to right: ground truth, frame 1, output, frame 3. (Best seen in color.)

Three random triplets have been chosen with average object displacement, small object displacement, and a variation of small and large objects displacement in addition to a moving background that has not been used before and was removed from the data set before initial training. All of these triplets have been compared to the different approaches discussed in Section 3.2.

Figure 13 shows small object displacements and a background that is reused (not redrawn for every frame) in the whole scene and translated slightly to the left throughout the sequence. This can be seen with (Input2, Different Input), where the background is not completely black and thus the background is moving within this sequence. This is interpolated well with the initial model and the double input model, because the background is very close in position within both input frames, which means less room to produce errors than with larger displacements. The overlay shows the characters left-hand movement, where the hands barely overlap. The double input model shows a color of the interpolated hand closer to frame 1 and frame 3 than the initial model. This is also the closest in the center of both hand positions. The 'different input' results shows a lot of noise and is inferior to the other approaches.

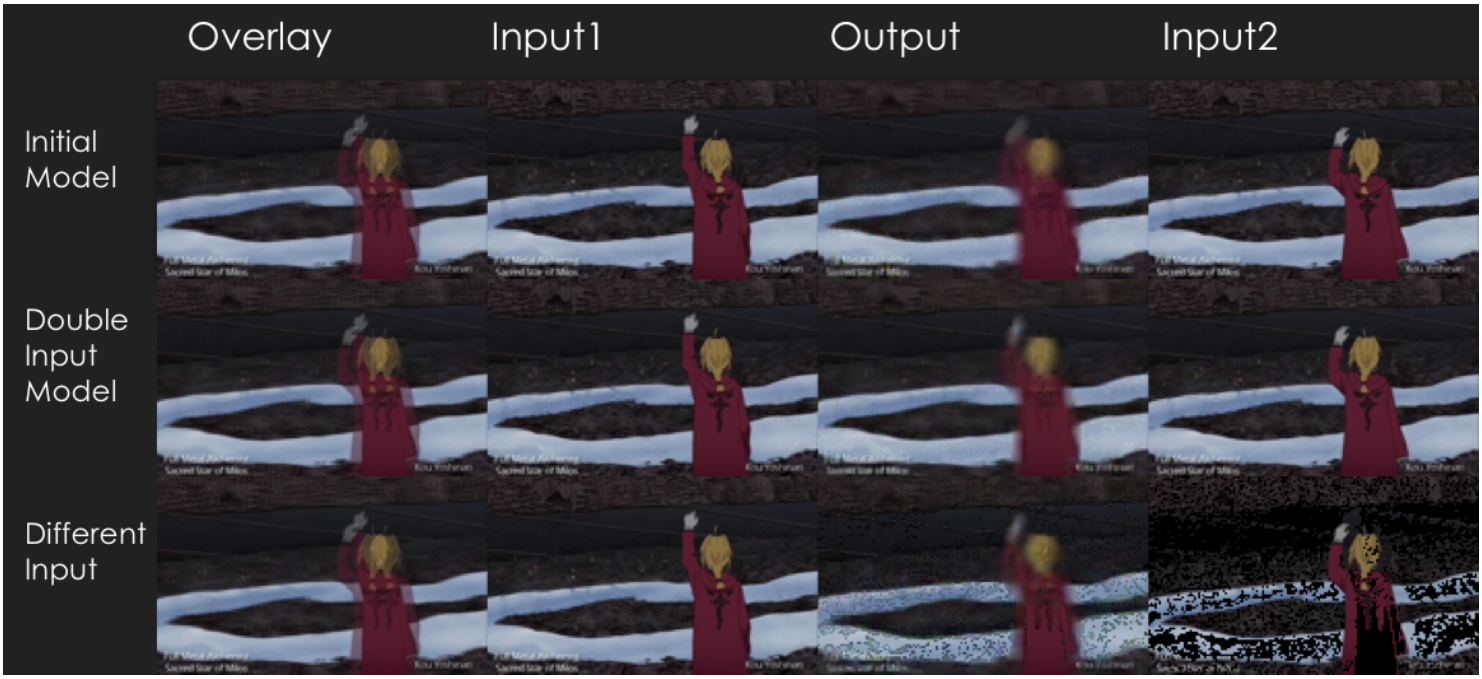

Figure 13: Animation example triplets, from left to right: ground truth, frame 1, output, frame 3

Figure 14 is an example triplet with average object displacement. Similarly to the triplet shown in Figure 13 we see her left hand moving in a particular way creating two hands really close to each other apparent in the overlay images. They do not overlap here either. Unlike the triplet with small displacement this triplet does not produce motion blur with the left hand

but rather two distinct separate blobs to replace them whereas Figure 13 showed one larger blur. This is likely due to the larger object displacement since the character is much larger in this triplet. When we look at the 'different input' approach we will notice that it is inferior to the other approaches again. At first glance, it looks like it contains sharper edges, but remember that pixels that are the same in both input frames are given priority in the output to produce better images as described in Section 3.4. These pixels have more contrast with the output which create the sharp edges.

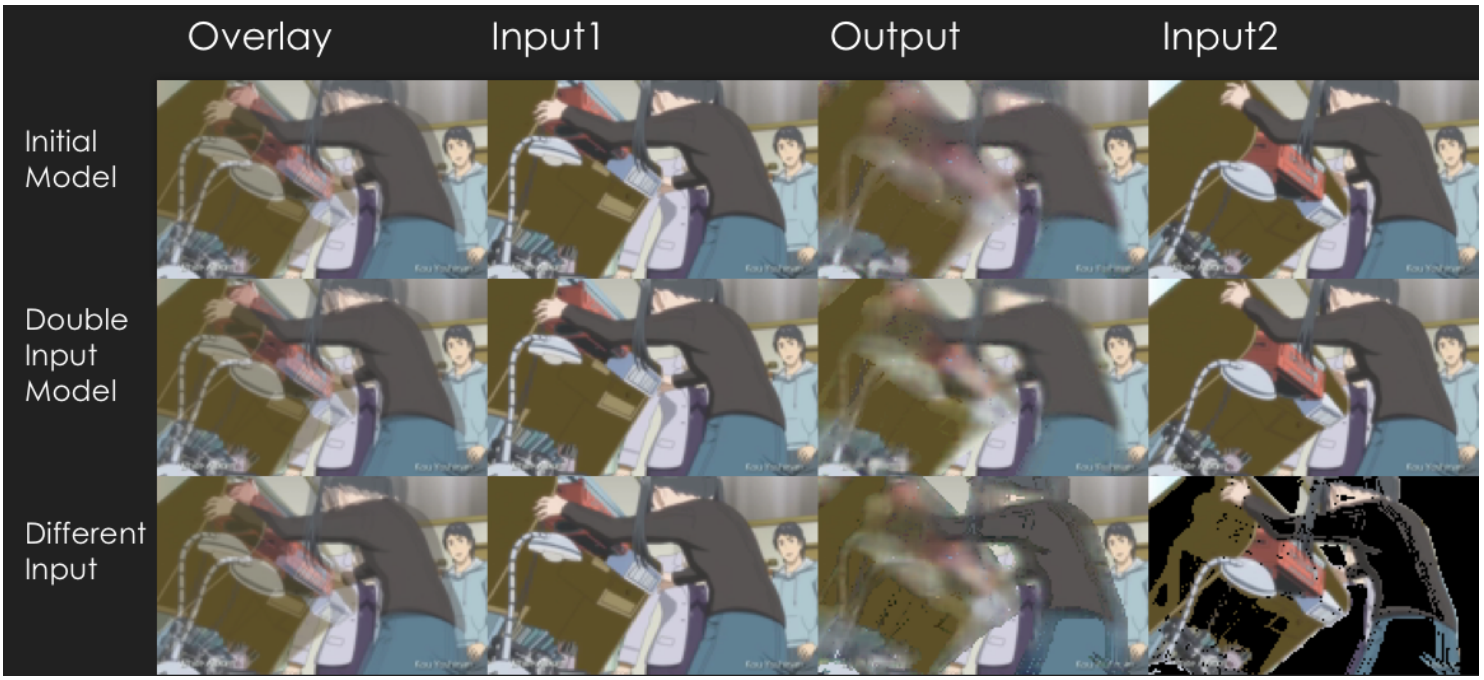

Figure 14: Animation example triplets with large displacement (Best seen in color.)

The triplet with both small displacement and large displacement is shown in Figure 15. The character shows little object displacement between frames and creates motion blur in the appropriate places. Look in particular at the pigtails which have slight overlap as seen in the overlay image. The initial model produces pigtails of similar size in the output whereas the double input model shows a slightly smaller volume compared to the input. The larger displacement shows the initial model blurring more than the double input model in the moving building.

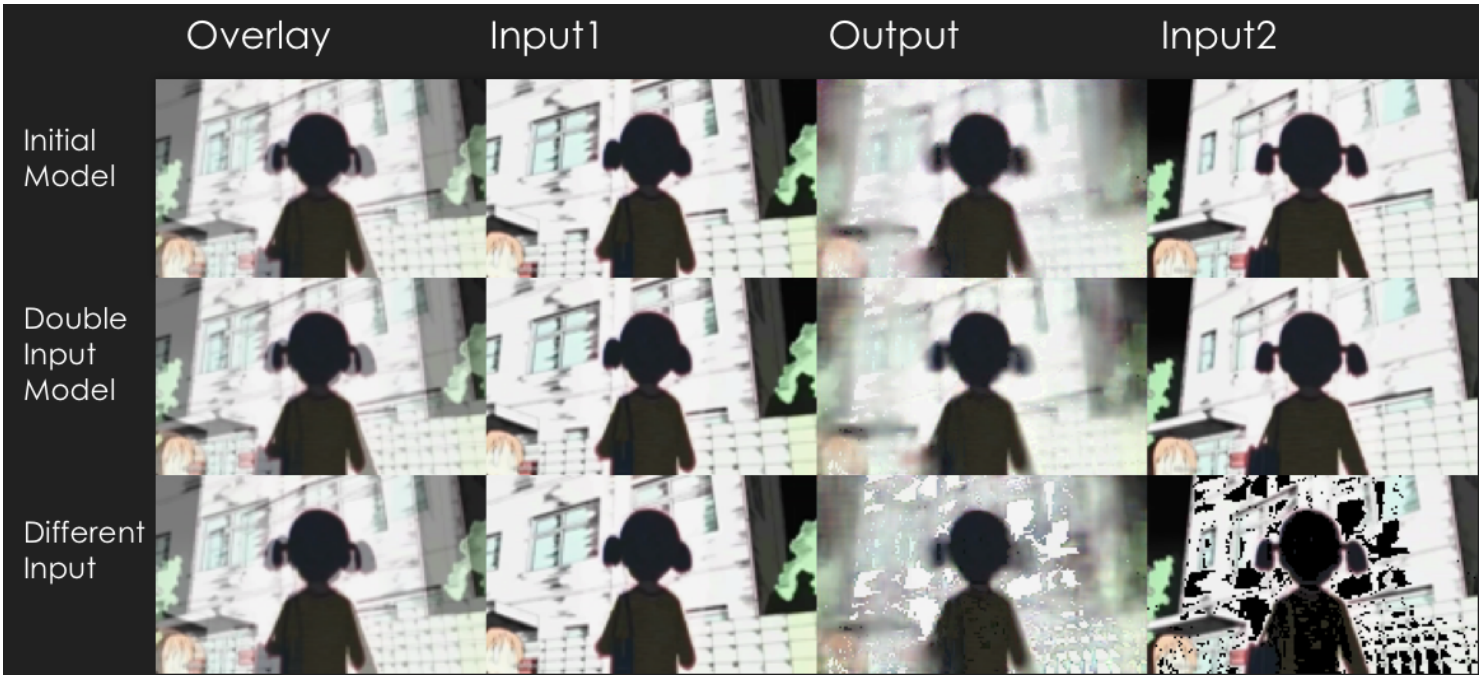

Figure 15: Animation example triplets, from left to right: ground truth, frame 1, output, frame 3

Overall the double input model and the initial model seem to outperform each other depending on the triplet, with no distinct features for when they outperform each other. This has been found by evaluating several other sequences. These slight improvements are also very minimal, which makes it hard to justify either performing better. These models could be very similar because they seem to adjust the weights in the double input model in a similar way as in the initial model, which is mainly to reconstruct the image and background as accurately as possible if they are the same.

4.3 Reconstructed Animation

The reconstructed animations with interpolated frames were compared to ones with an overlay of their neighbouring frames and the ground truth animations. The ground truth is qualitatively better compared to the overlay-and interpolated animations and shows a clear difference in terms of motion and consistency. The blurriness produced by the interpolated frames causes its animation to seem like it is flickering. The overlay animation also contains some flickering, but contains much sharper images and has been found to be more preferred. This conclusion is based on a small survey with 21 participants, in which 3 scenes constructed with CNN output every other frame were compared against the same 3 scenes reconstructed with overlay images instead. The ground truth animation was not shown, because the difference between these animations and the overlay/CNN animations was clearly visible. This would potentially make it hard to see the differences between the overlay and the CNN animations due to their closer similarity.

The 3 scenes[7] show the reconstructed animation with the overlay and with the double input model side by side, but alternating between left and right for each scene. The order of the methods was not disclosed during the survey. They were asked to pick their preferred method by stating *'left'* or *'right'* and focus on the movement and overall pleasantness of the animation for each scene. The preferences were counted and the final measurement was the method chosen the most for each individual participant. Initially little difference between the

[7]This video was not used for the survey but contains the same scenes with the overlay on the left and the CNN version on the right (as of June 2016): https://www.youtube.com/watch?v=MODbh6M-g00

methods was noted by many, but 20 out of 21 participants preferred the overlay scenes more than the CNN after watching the scenes for a maximum of 3 times in the end. Their reasoning was that the blurriness caused that method to seem more unpleasant than the overlay.

5 Conclusion

The question of whether it is possible to generate frames automatically in the domain of 2D animation using CNNs is a question that will be answered through a continuing process of improving CNNs over time. With the methods described in this thesis, it is at its early stages. The results of the animation show blurry interpolated frames and are at their best when there is little object displacement. This has been achieved by trying out variations of existing models. Frame interpolation shows promising results with different video data sets other than KITTI RAW in Long et al.[5] as well, which lead to the conclusion that the animation data set is too difficult for the variations of the model tested in this thesis.

5.1 Discussion

The difference between the animation data set and the KITTI RAW data set is that there is not a consistent motion in all of the animation scenes. The car that provided motion in the KITTI RAW data set drove at a relatively consistent speed throughout the scenes whereas the animation data set does not contain this consistent speed. KITTI RAW also contains images of particular subjects such as a city road, where the animation data set scenes vary vastly from each other and can range from a character in mid-conversation to flying vehicles.

The residual connections argued in Section 3.2.3 to retain fine-grained details might also be the cause for poor frame interpolation. The model could see these residual connections as an easy way to significantly reduce the overall error, which would mean it would get stuck in a local optimum. In this case, the innermost convolutional layers would essentially be disregarded as these features have lost most fine-grained information to reconstruct the input.

6 Future works

6.1 Data set

The CNN also had to learn to recreate the background, where this is typically just copied when creating animation. Special effects and lighting effects also introduce a lot of noise, and this is not done when animating characters but only after initial coloring and in-betweening. Animating characters and objects typically happen by creating them from scratch, while special effects, lighting and backgrounds can be reused throughout the scene. Frame interpolation could be more effective when done on images that do not contain this.

Thus a more effective data set could contain images without backgrounds and no lighting effects, but just flat colors within the moving objects. This way frame interpolation could be applied more concisely, and special effects, lighting and backgrounds would not be of influence.

6.2 Deep matching

Weinzaepfel et al.[8] showed a model that is similar to CNNs where optical flow estimation is done with supporting data of matching image chunks in two different images. This matching algorithm is named *deep matching*. The additional data worked effectively for larger displacements in their model and could prove useful as additional data for a CNN as well.

6.3 Loss function

The loss function should be more applicable to the problem, where smaller object sizes should be punished as equally as larger objects. Recall Section 3.3.1 where the problem was introduced. Additionally the L1 error could be divided by the total volume of pixels where the values differ in the whole triplet, this way the object displacements are normalized for size differences. The loss function should also punish blurry patches more. Color values are largely consistent because of single color fills when using a data set that does not contain lighting, special effects or a background. The number of different values can be compared to the number of different values in the ground truth and could possibly be used in the loss function if this is differentiable.

6.4 Optical Flow Estimation

Research on improving optical flow estimation will help advance frame interpolation in general. This could in turn help frame interpolation for 2D animation. CNNs are improving at a rapid rate at computer vision tasks, while they become computationally cheaper to perform. This usually results in different models, which animation frame interpolation models could inherit from.

References

[1] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.

[2] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] Long, Kneip, Alvarez, and Li. Learning image matching by simply watching video. 2016.

[6] Zimmer Grosse Meyer, Wang. Phase-based frame interpolation for video. 2015.

[7] Damien Teney and Martial Hebert. Learning to extract motion from videos in convolutional neural networks. *CoRR*, abs/1601.07532, 2016.

[8] Philippe Weinzaepfel, Jérôme Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV 2013 - IEEE International Conference on Computer Vision*, pages 1385–1392, Sydney, Australia, December 2013. IEEE.

Appendices

A Code

All the code can be found here https://github.com/heytemb/Frame-Interpolation