BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Generative image inpainting enhanced with edge focused WGAN-GP adversarial loss.

Authors: Tomáš Beránek (xberan46), Richard Klem (xklemr00)

June 4, 2022

1 Abstract

We improved the original Yu's [5] model by adding a third discriminator. The original solution uses the local discriminator to calculate the loss on the generated patch and the global discriminator to calculate a loss on the whole image. But very often, we can see the edges of the patch, because there is no special attention to smooth transition between the generated patch and the rest of the image. We decided to provide a new discriminator, which will be focusing on the patch with the near surroundings. This should help the generator to learn how to make better transitions on the edges of the patch and it should also lead to more clear/sharp patch.

2 Related work

According to Jiahui Yu [5], there are two main categories. The first is a diffusion-based or patch-based approach and the second one is a learning-based or attention modeling approach. The diffusion-based model uses variational algorithms to propagate information from the background regions to the holes. Patch-based approaches use patch similarity to do so. Diffusion and patch-based approaches work well for stationary textures as you can see in figure 1. The image on the right is generated by joint interpolation of the image gray levels and gradient/isophotes directions. [1] On the other side, these methods have decreased the quality of output for non-stationary data such as natural images.

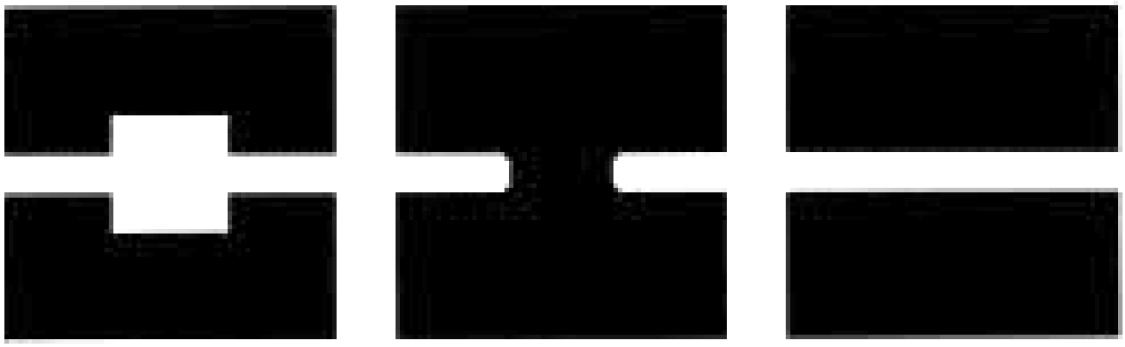

Figure 1: On the left side: the input image with square mask, on the right side: the result from joint interpolation method by Ballester et al.

3 Dataset

This section describes both datasets from the original work and what we used in our work. Both we, and Yu and al. use Places2 dataset. [1] More specifically the high-resolution images, which have been resized to have a minimum dimension of 512 while preserving the aspect ratio of the image. Original images that had a dimension smaller than 512 have been left unchanged.

For time reasons, we decided to use just a subset of the Places2 dataset. We reduced our dataset from the original 365 categories to the 7 categories: *army_base*, *badlands*, *corn_field*, *forest_path*, *golf_course*, *hayfield* and *mountain*. Each category has always 5000 images. We split it in a certain ratio, so we have a training dataset containing 4500 images and a testing dataset containing 500 images from each category. It makes 35 thousand pictures in total. 31 500 images serve as a training set. 3500 pictures were selected as testing/validation set.

[1] http://places2.csail.mit.edu/

4 The original work and article

The Yu's [5] solution is based on the work of [2] named *Globally and locally consistent image completion*. Yu et al. improved this approach with various changes and additions. In figure 2 you can see the architecture of the Iizuka et al. model. It is worth mentioning for future reference that the completion network is one piece network and that the outputs of local and global discriminators are merged.

The one of the main changes is split of completion network into two stages – coarse-to-fine network. Coarse network (the first stage) makes an initial coarse prediction. Then the refinement network (the second stage) takes the coarse prediction as input and predict refined result. Also coarse and refinement networks are trained with different combinations of loss functions. The coarse network is trained with the reconstruction loss explicitly, while the refinement network is trained with the reconstruction loss as well as GAN losses. They also removed the batch normalization layers, which they found deteriorates color coherence.

In the Iizuka's [2] work is used concatenation layer, which concatenates outputs from local and global discriminator. But Yu et al. found, that it is better if they keep the outputs from discriminators separate. Regarding GAN architecture, in the Yu's work, they used Wasserstein GANs as opposed to DSGANs used in Iizuka's work. More specifically, Yu et al. used WGAN-GP. They also stated that WGAN-GP loss works well when combined with ℓ_1 reconstruction loss as they both use the ℓ_1 distance metric.

To prevent pixel hallucination and very different content in the center of the generated patch, Yu et al. introduce spatially discounted reconstruction loss using a weight mask. The weight of each pixel in the mask is computed as γ^l, where l is the distance of the pixel to the nearest known (not masked) pixel. γ is set to 0.99 in all experiments. They use discounted ℓ_1 reconstruction loss in their implementation.

All these changes and improvements made by Yu et al. leads to faster convergence (as far as we can speak about convergence in generative models) than the Iizuku's work. Also it resulted in more accurate inpainting outputs. Yu stated a 100x speedup, nevertheless we personally take it with a grain of salt as the proposed numbers does not correspond well with the Iizuka's paper.

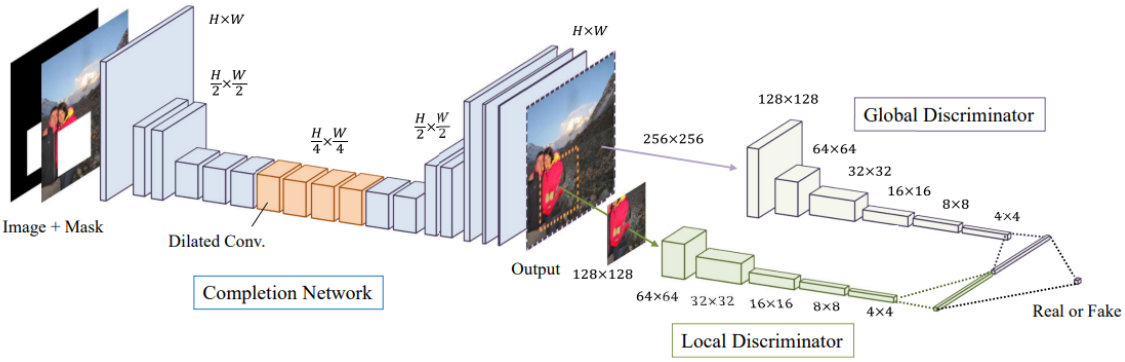

Figure 2: Architecture proposed by Iizuka et al. based on one completion generative network and two discriminators.

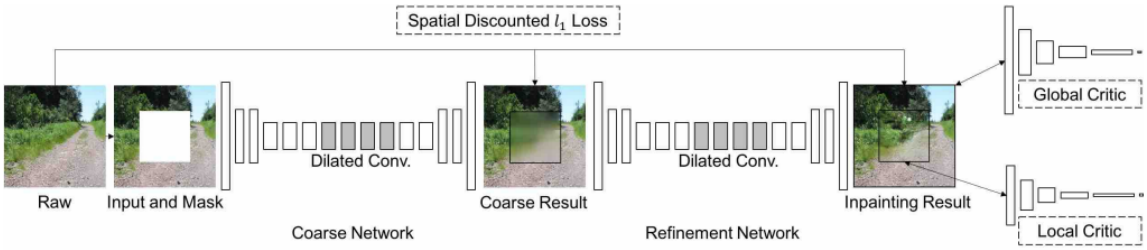

Figure 3: Architecture modified by Yu et al. You can see generative network split to coarse and refinement network and discriminators kept separated.

5 Our enhanced solution – edge focused discriminator

We have noticed that Yu's [5] results are high quality, but there is often observable transition between the generated patch and the rest of the image. We were thinking about how to improve this aspect and we thought we could add another discriminator, which will be focused on the edges of the patch and the near surrounding. We want to smooth the transition between the patch and the rest of the image and also improve the sharpness of the patch. For these purposes we used the Sobel edge detector. After extracting the edges, it should be easier for the discriminator to recognize the *fake* image because the sharp transition between the patch and the surroundings creates a fairly clear edge. And if the generated patch is very blurry, it will contain fewer edges than the surrounding patch, again making identification easier. Intuitively: to compensate for these properties, the generator is forced to make a smooth transition and generate more sharp edges.

To detect edges in the patch we choose to use Sobel edge detection[2]. This method is able to detect both vertical and horizontal edges in the image. It is detecting all edges in the input image, so we had to create special input for this method to successfully use it for our use-case. First we convert image to the gray-scale and then the Sobel edge detection is applied. The ouput of the Sobel edge detector is an image with the same dimensions but each pixel is converted to the white when it is considered as an edge or black if it is not. How the Sobel edge detection works can be seen in figure 4 which has been taken from Verma et al. work [3].

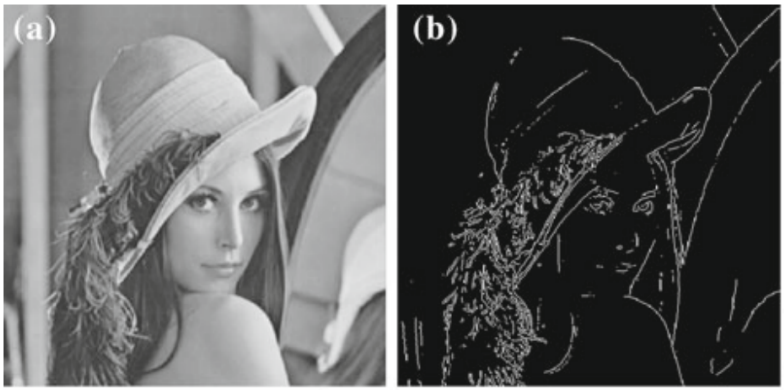

Figure 4: a) The original image converted to the gray-scale, b) Output from Sobel edge detection.[3]

The Sobel edge detection is part of the edges discriminator. Similarly to the other two discriminators, the output of the Sobel edge detection is followed by a few convolution layers,

[2] https://www.tensorflow.org/api_docs/python/tf/image/sobel_edges?authuser=1

which results to binary classification – *real or fake*. The Sobel edge detection can be understood as some pre-processing.

For the ability to a proper comparison of Yu's model with our improved model, we decided to train the model with the same settings as Yu et al. That is why we use a random selection of the input image cropped to 256x256 pixels. For example, if the input image has a size of 734x512 pixels, the algorithm randomly selects an area of 256x256 pixels. In that cropped input image, there is randomly cut off the mask of the size of 128x128 pixels. While creating the input data (referenced as *edges patch* furthermore) for the edges discriminator, we took the local patch but with an extra 16 pixels added in each direction. The final size of this edges patch is therefore +32 pixels in both height and width. For our particular case, it means 160x160 pixels (128+32=160). You can see our improved architecture in figure 5

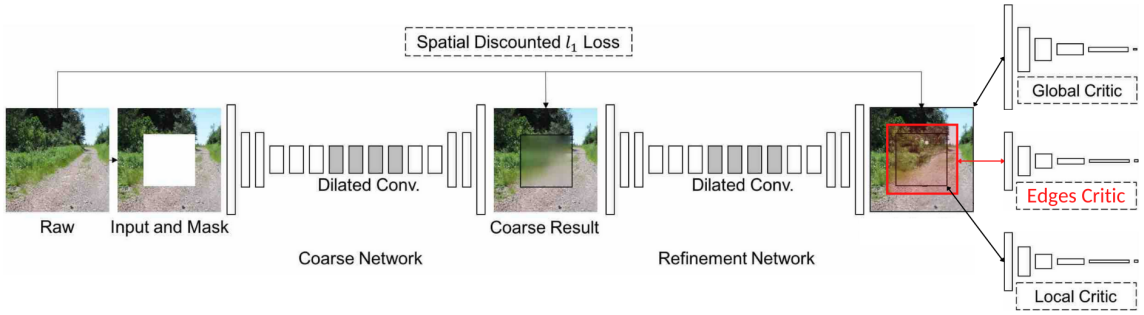

Figure 5: Our improved architecture with extra discriminator focus on edges of the generated patch and the near surroundings.

Yu's solution uses a simple sum of losses from the global and local discriminator. We followed the same approach and we have added loss from edges discriminator to the previous ones. Regards the weights they are the same, set to $\alpha = 1.0$. So the formula for the overall loss from all three WGAN-GP discriminators is $losses = 1.0 * loss_{global} + 1.0 * loss_{local} + 1.0 * loss_{edges}$.

The rest of the logic of the model remained untouched. That means we did not change any part of the generator nor any of the original discriminators. We have enhanced what was already working well.

6 Training, testing and validation

We have trained both the baseline and our modified model from scratch, so we can do a meaningful comparison. Because we were training on our machines, we had to modify the original batch size from 16 to 8 images. We trained both the original and our model for 100,000 iterations. We set the training process to save checkpoint models during the training process, so we can do a comparison if the generator is improving along with the training. As already discussed in section 3 we have trained on seven categories, with 4500 images each category.

The validation and testing of generative models are very tricky and difficult. There are many methods to evaluate the technical properties of an image, but it is difficult to use them to validate that generator is generating some meaningful human-eye pleasant output. The generator could produce nonsense, but with nice SNR, and then it will be evaluated as a good quality image.

Since one of our goals was to try to generate more sharp patches, it was necessary to introduce a quantitative measurement of image sharpness. As a metric, the average value of the gradient of all pixels was used. The sharper the edge, the larger the values between adjacent pixels – this can be expressed as a gradient. Gradients were calculated in the vertical and

horizontal directions. The Euclidean distance of these values was then the sharpness value for a particular pixel. The results of the sharpness measurements on 700 images (only on the generated patches) are:

- 9.802 for our model,
- 9.546 for Yu's model.

The most accurate measurement if the image looks nice and natural is manual testing. We were able to manage several manual testing sessions. The task of this manual testing was choosing between two images, which one did the users like more. If the users were not sure where are the differences, the users could show the mask so they can focus on that/these specific parts of the images. Overall we collected 629 comparisons from 6 unique people and 177 unique images. The results are the following:

- 67.23% (119/177) of images from our model were voted as better compared to the 32.77% (58/177) from Yu's model.
- 66.61% (419/629) of votes were for our model compared to the 33.39% (210/629) for Yu's model.

7 Experiments

We did some experiments with activation functions in the generator. We tried to use Swish instead of ELU according to the blog article of Andre Ye[4]. The Swish activation function should be the very best alternative to ReLU, but according to the experiments proposed in the article, Swish should also outperform ELU in eight of nine cases. But unfortunately for us, we cannot confirm this in our particular use case. The output of the generator was not better, because it creates a style of *"oil in the water puddle"*, which was not desired. You can see the swish properties in the figure 6.

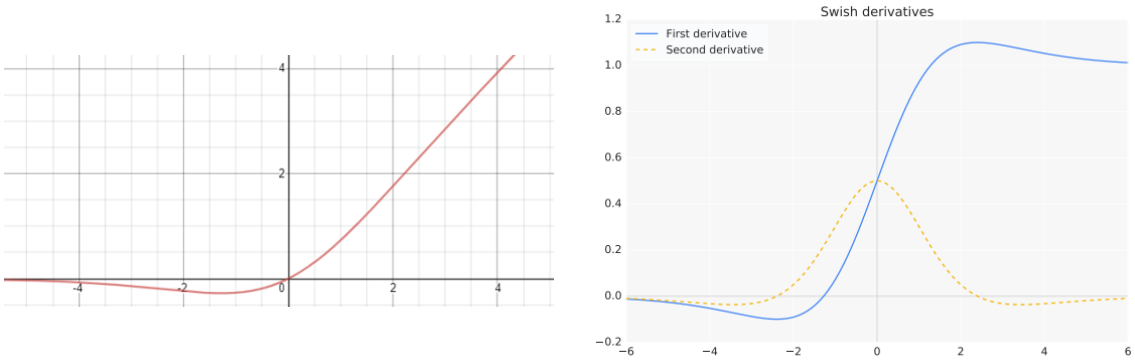

Figure 6: Swish activation function properties. Taken from Ye's work.[4]

For furthermore development, there is plenty of space. Some other, maybe more advanced/accurate methods of edge detection could be used. In the original Yu's solution there is used ReLU at one place, which is the end of convolution layers series in contextual attention. We expect, that in this specific case the ReLU could be replaced with the swish function with success. Another experiment can be done by normalizing discriminators' loss sum. The original solution uses the same $\alpha = 1$ and we kept to this. But it could be interesting to experiment with $\alpha = \frac{2}{3}$ or so.

8 Results and conclusions

We came up with an improvement of the state-of-the-art approach. We managed to implement this enhancement successfully. We have done some experiments and stated what are other possibilities on how to improve this model. Also, we have done extensive manual testing with multiple users and plenty of example images for both models and we evaluated that our improved model generates about twice as many human-eye pleasant images as the original one. According to the manual testing results, image sharpness analysis, and based on provided examples below, we can state, that we have enhanced the smoothness of the transition between the edges of the patch and the rest of the image. There are some examples in which it can be seen the difference between the original and our model. For each example there is at first the mask, the second is the output of Yu's model, and then third, the last one is our output. There is the mask for the reason, we cannot mark a bounding box in a red rectangle for example as we want to show the edges of the patch.

Figure 7: Example n.1 – mask.

Figure 8: Example n.1 – Yu (baseline).

Figure 9: Example n.1 – ours.

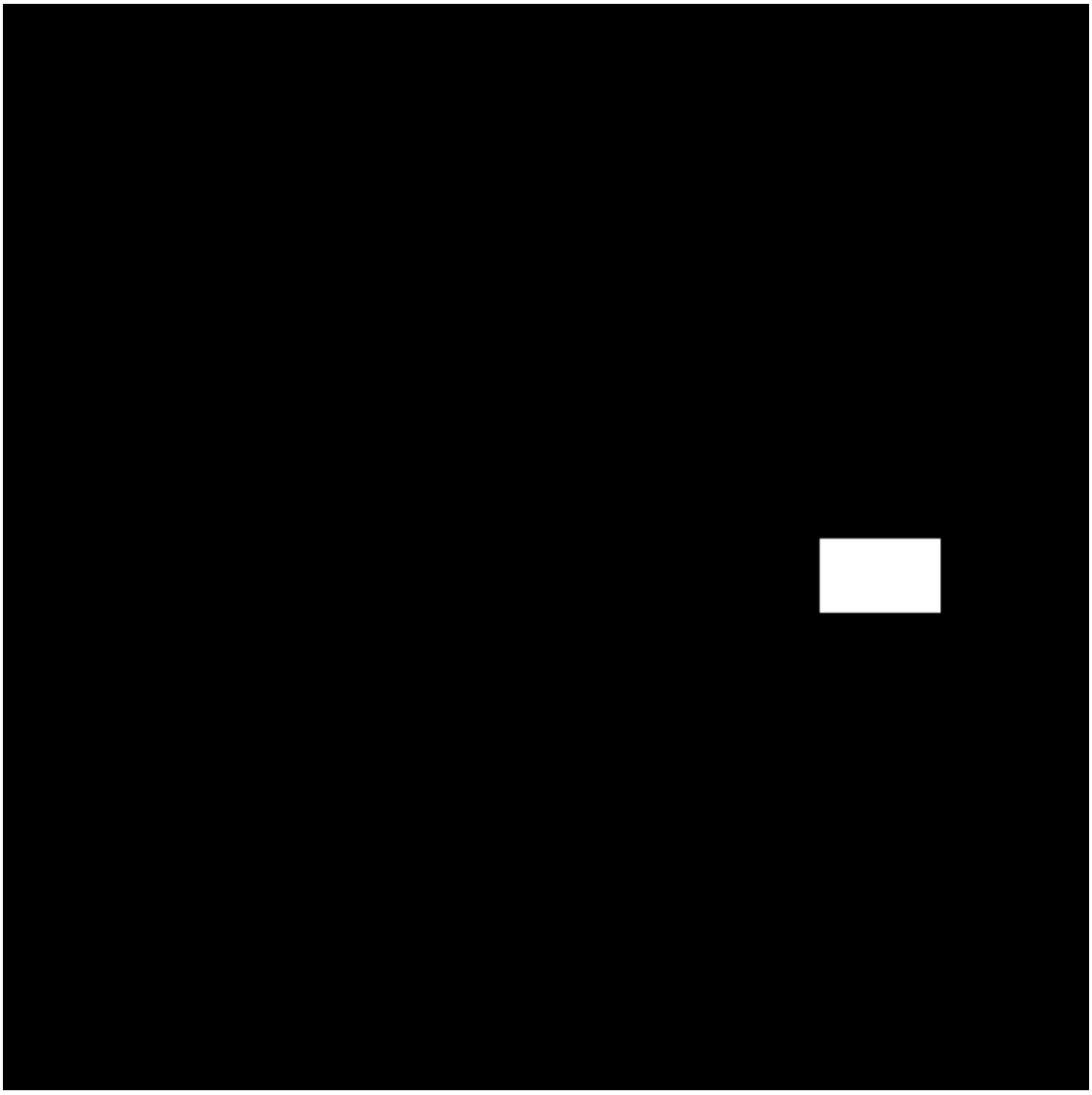

Figure 10: Example n.2 – mask.

Figure 11: Example n.2 – Yu (baseline).

Figure 12: Example n.2 – ours.

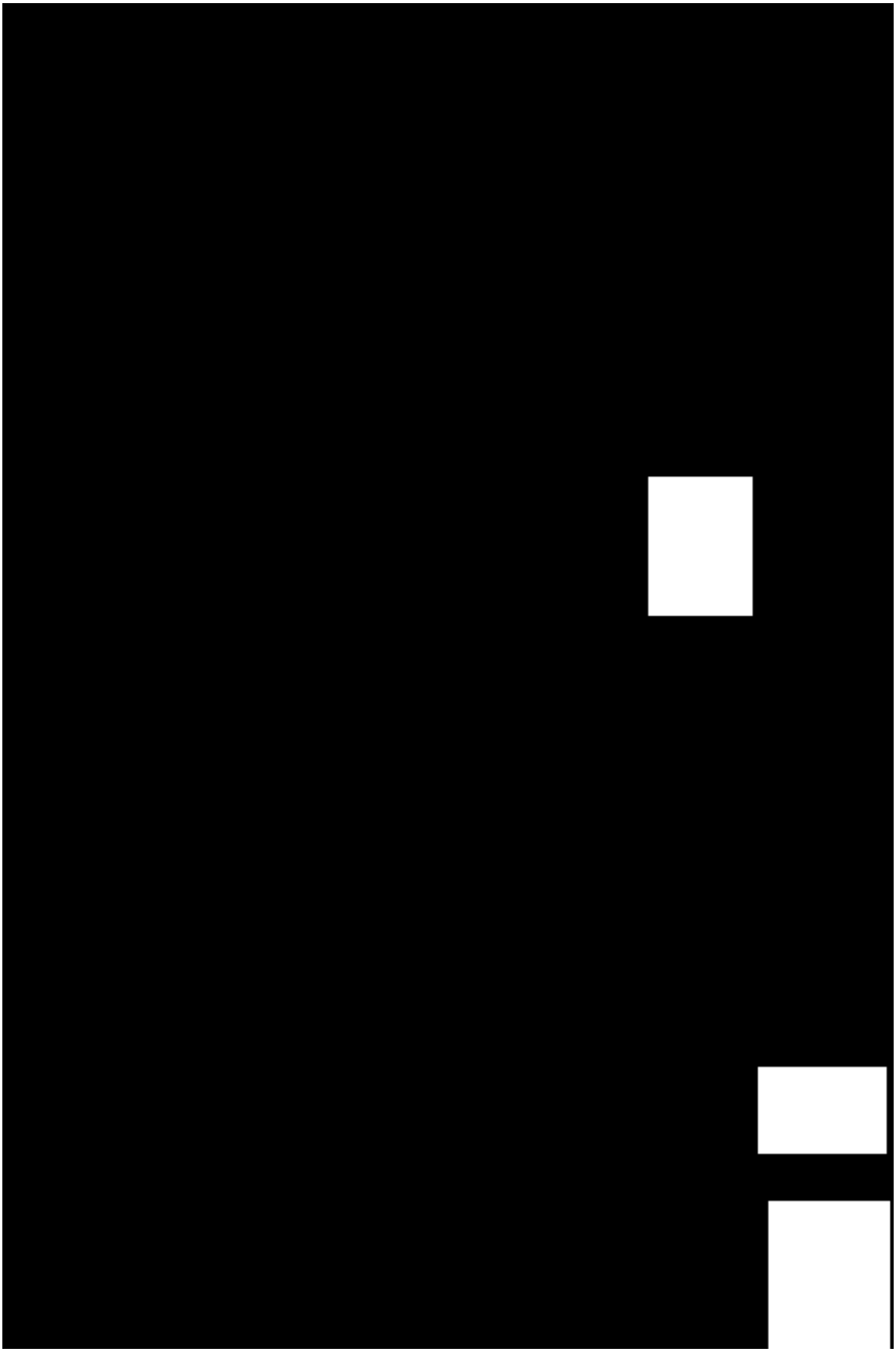

Figure 13: Example n.3 – mask.

Figure 14: Example n.3 – Yu (baseline).

Figure 15: Example n.3 – ours.

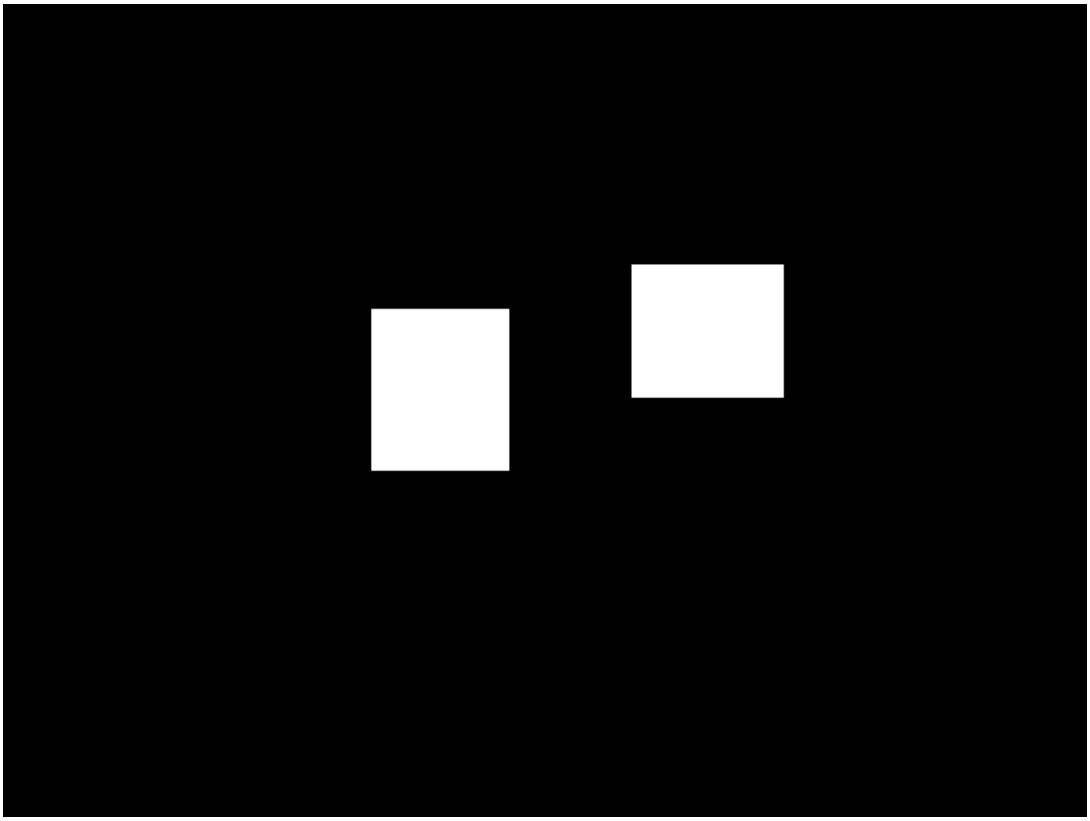

Figure 16: Example n.4 – mask.

Figure 17: Example n.4 – Yu (baseline).

Figure 18: Example n.4 – ours.

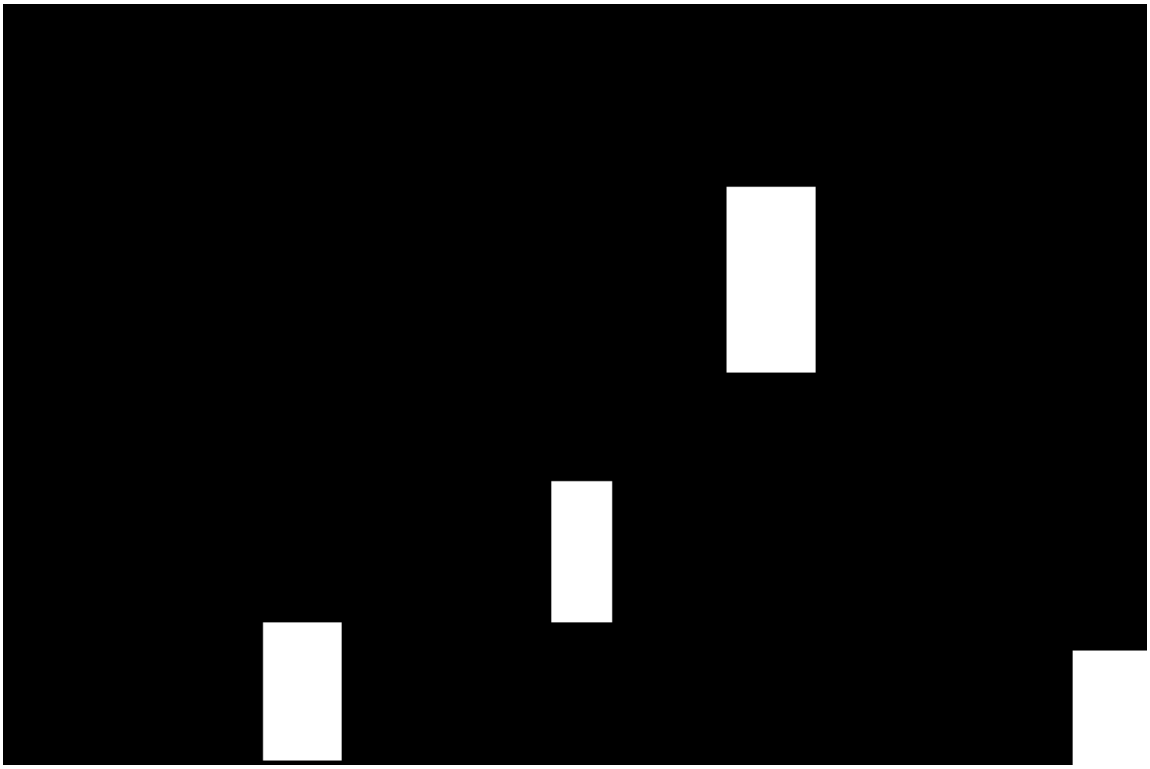

Figure 19: Example n.5 – mask.

Figure 20: Example n.5 – Yu (baseline).

Figure 21: Example n.5 – ours.

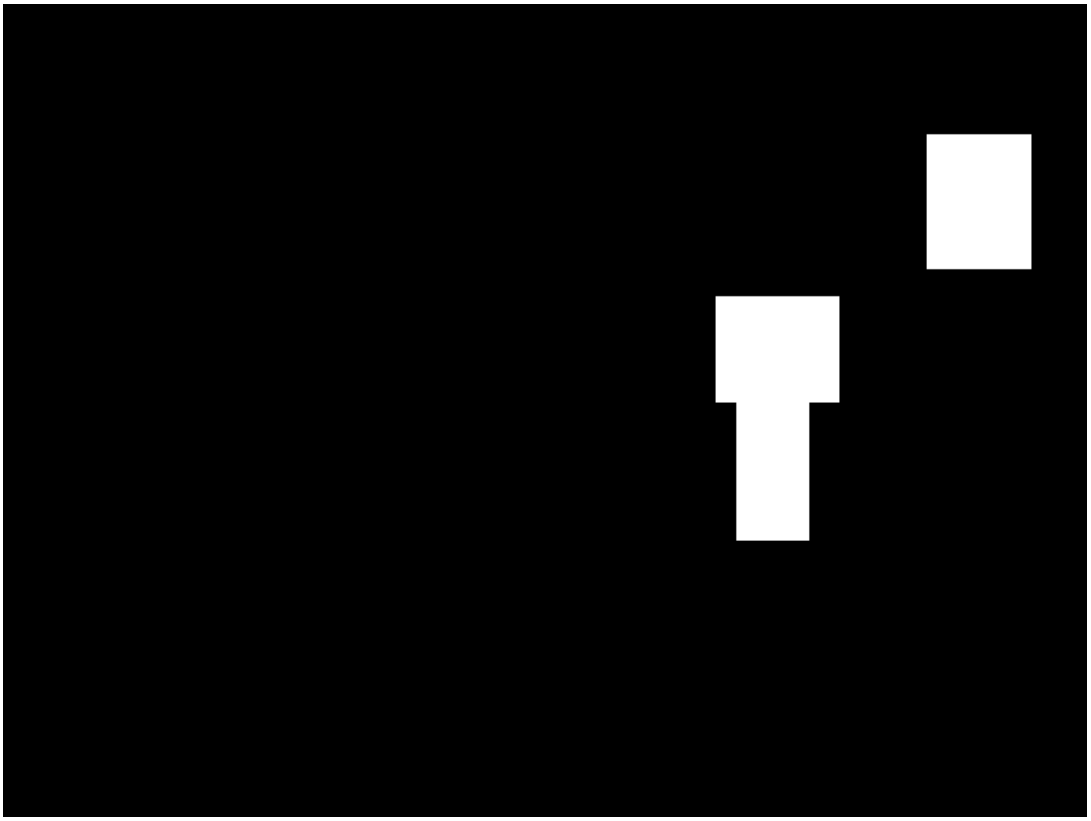

Figure 22: Example n.6 – mask.

Figure 23: Example n.6 – Yu (baseline).

Figure 24: Example n.6 – ours.

Figure 25: Example n.7 – mask.

Figure 26: Example n.7 – Yu (baseline).

Figure 27: Example n.7 – ours.

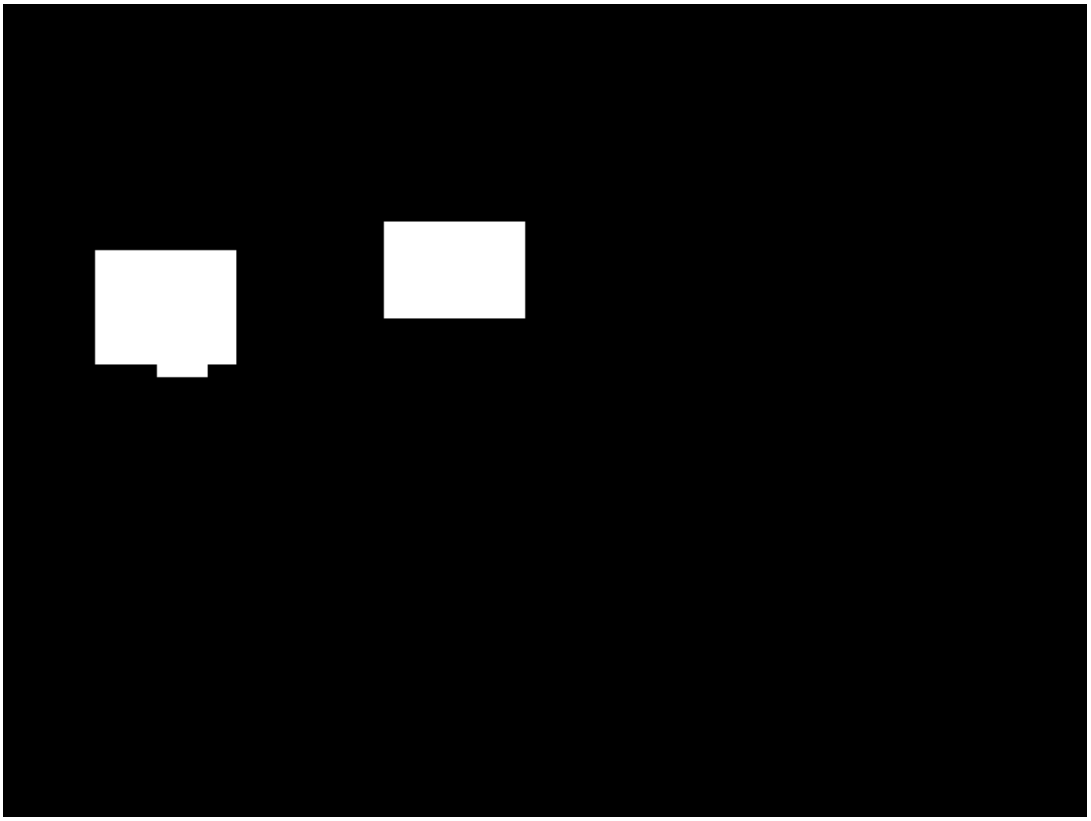

Figure 28: Example n.8 – mask.

Figure 29: Example n.8 – Yu (baseline).

Figure 30: Example n.8 – ours.

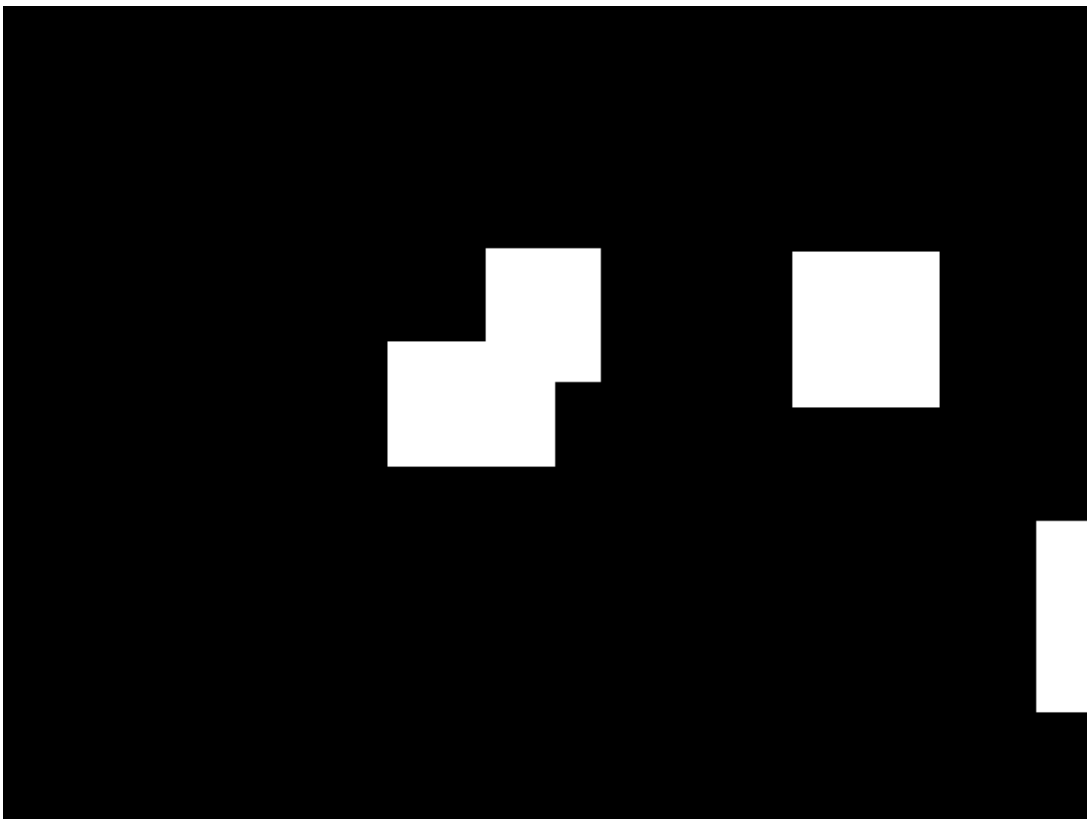

Figure 31: Example n.9 – mask.

Figure 32: Example n.9 – Yu (baseline).

Figure 33: Example n.9 – ours.

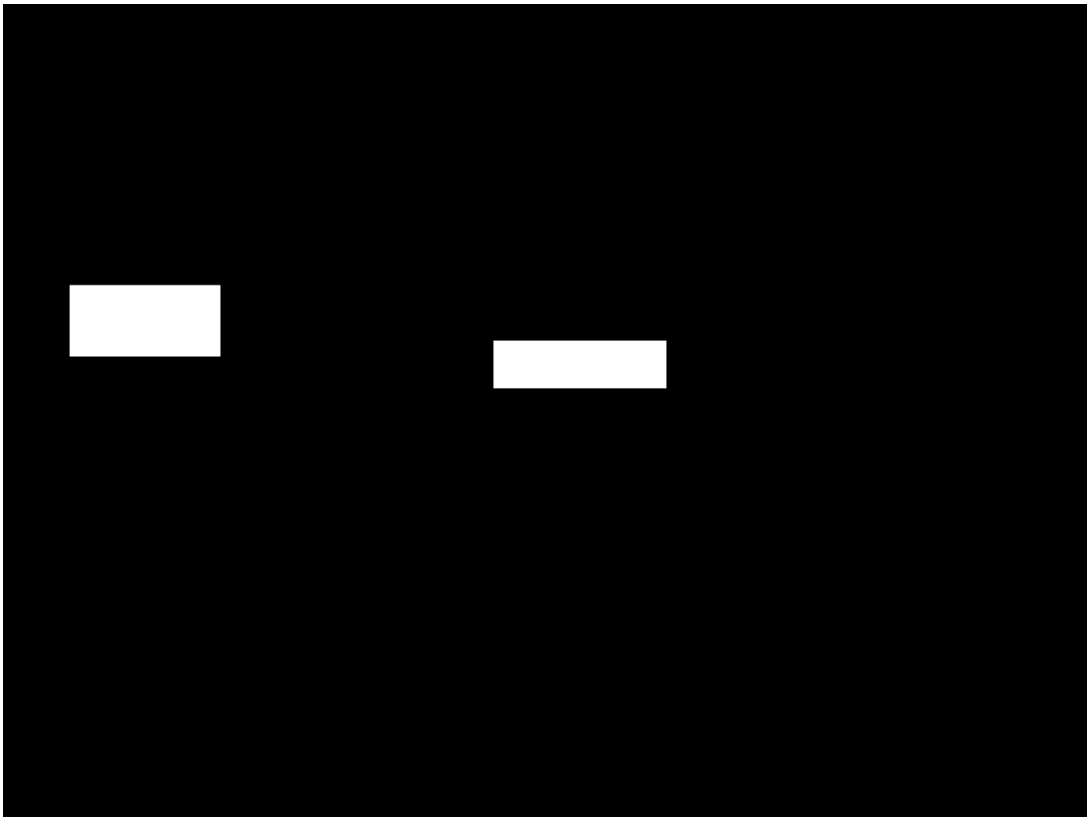

Figure 34: Example n.10 – mask.

Figure 35: Example n.10 – Yu (baseline).

Figure 36: Example n.10 – ours.

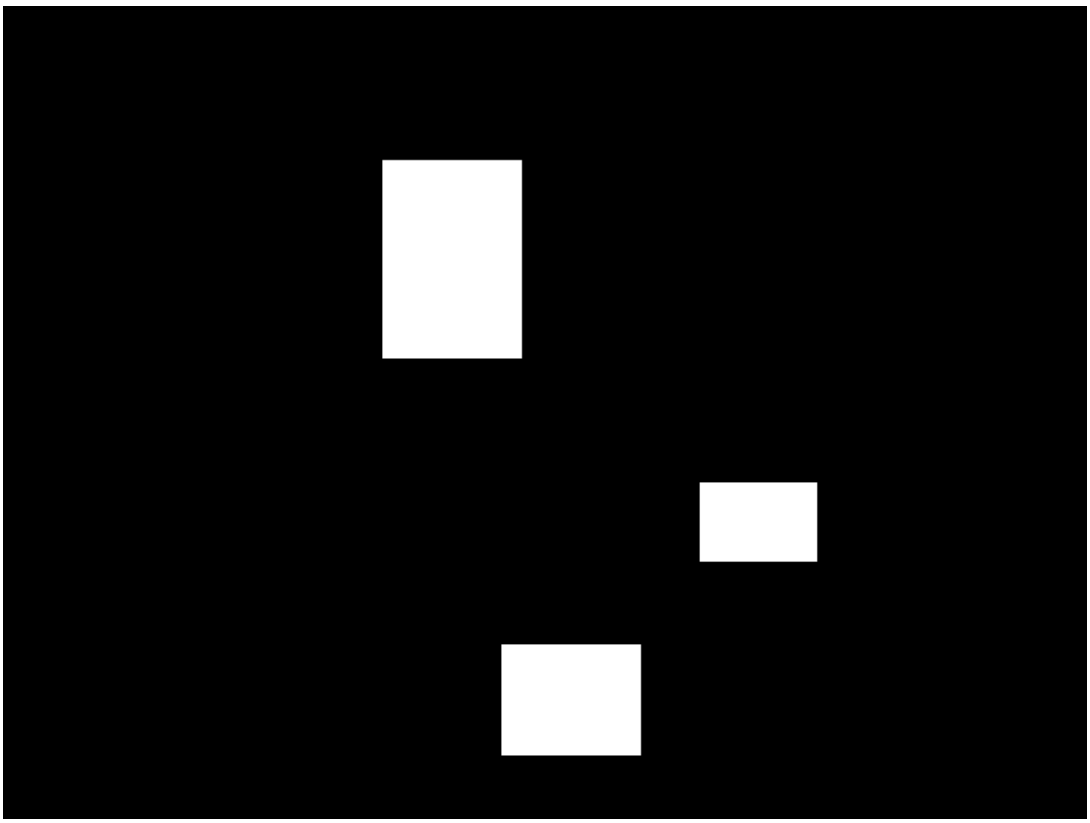

Figure 37: Example n.11 – mask.

Figure 38: Example n.11 – Yu (baseline).

Figure 39: Example n.11 – ours.

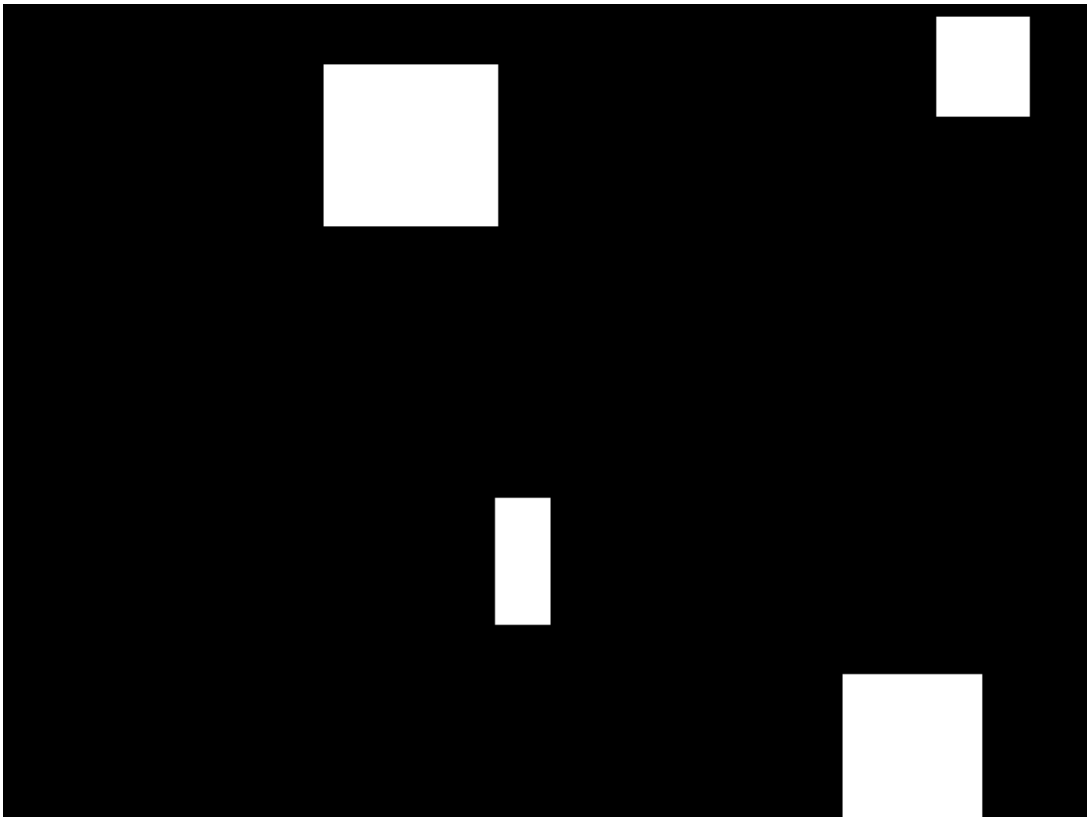

Figure 40: Example n.12 – mask.

Figure 41: Example n.12 – Yu (baseline).

Figure 42: Example n.12 – ours.

References

[1] BALLESTER, C., BERTALMIO, M., CASELLES, V., SAPIRO, G. and VERDERA, J. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Transactions on Image Processing*. 2001, vol. 10, no. 8, p. 1200–1211. DOI: 10.1109/83.935036.

[2] IIZUKA, S., SIMO SERRA, E. and ISHIKAWA, H. Globally and Locally Consistent Image Completion. *ACM Trans. Graph.* New York, NY, USA: Association for Computing Machinery. jul 2017, vol. 36, no. 4. DOI: 10.1145/3072959.3073659. ISSN 0730-0301. Available at: https://doi.org/10.1145/3072959.3073659.

[3] VERMA, O., HANMANDLU, M., SULTANIA, A. and PARIHAR, A. A novel fuzzy system for edge detection in noisy image using bacterial foraging. *Multidimensional Systems and Signal Processing*. march 2013, vol. 24, p. 18. DOI: 10.1007/s11045-011-0164-1.

[4] YE, A. *Swish: Booting Relu from the Activation Function Throne*. Towards Data Science, Mar 2020. Available at: https://towardsdatascience.com/swish-booting-relu-from-the-activation-function-throne-78f87e5ab6eb.

[5] YU, J., LIN, Z., YANG, J., SHEN, X., LU, X. et al. Generative Image Inpainting with Contextual Attention. *ArXiv preprint arXiv:1801.07892*. 2018. DOI: 10.48550/ARXIV.1801.07892. Available at: https://arxiv.org/abs/1801.07892.