Video Inpainting - Removing objects
Final Project Report

Loïck Chambon
MVA - ENS Paris Saclay
loick.chambon@eleves.enpc.fr

Abstract

Video inpainting is the task of reconstructing missing pixels in a video. It is an important problem in computer vision and an essential feature in many imaging and graphics applications, e.g. object removal, image restoration, manipulation, retargeting, image composition and rendering. While image inpainting is an almost solved problem, video inpainting is more difficult to solve as approaches are often unable to maintain the sharpness of the edges and create blurry effects while being unable to remove the correlation effect of an object. In addition, some suffer from temporal coherence. Although modern approaches overcome some of these problems, most of them require a complex input mask, cannot handle multiple deletion and are unable to remove correlations associated with an object. Recently, a paper has found a new way to combine objects and their effects to create masks containing subjects and effects in a self-supervised manner using only masks and coarse segmentation images. It does this by decomposing a video into a set of RGBA layers representing the appearance of different objects and their effects in the video. Although this requires training one model per video, it can lead to many applications.

1. Introduction / motivation

The purpose of inpainting is to fill in missing regions of an image. It has many real-world applications, such as removing unwanted objects, restoring, duplicating or enhancing objects in a image. The basic approaches are to search for the most similar image patches from the remaining pixels of the image itself or from a large dataset containing millions of images. However, the search algorithm can be time consuming and involves elaborate hand-crafted distance measurement. More recent approaches use GANs even with a single image [10] or a context encoder [3]. This topic has been widely studied in the case of images, to the extent that commercial image inpainting products for the general public are available, such as Photoshop's "Content Aware fill" or Google Pixel 6's "Magic Eraser".
The main motivation for this research is that video inpainting is a natural extension that can lead to many applications, software and products. It has many important and useful applications such as film restoration, professional post-production in cinema and video editing for personal use. However, it is an open problem because it requires maintaining the temporal consistency of the filling pixels. There are many difficulties: dealing with multiple moving objects, taking into account camera movement, removing correlations between objects, correctly reconstructing dynamic textures, etc. But, what is the point of removing an object from a photograph if its shadow still indicates its presence?
In this report, we will focus on a recent method called Omnimatte [5] that is able to automatically find the effects of several objects within videos and remove them. To take a critical look at our network, we will use real videos and compare our network with another method [7].

2. Related work

Generally speaking, former video inpainting algorithms belong to either the "patch-based" or "flow-based" category. Patch-based are based on the intuitive idea of copying and pasting small video "patches" into the occluded area. Nearly methods come naturally from the solutions proposed for image inpainting. But, in practice they suffer from time coherence and have some difficulties to handle camera motion as presented by Patwardhan [1]. To address the global inconsistency issue, patch-based completion algorithms have been cast as a global optimization problem that is minimized by alternating between patch search and reconstruction steps [14]. In contrast, flow-based methods jointly synthesise colour and flow and propagate colour along flow trajectories to improve temporal coherence, resulting in a more consistent result. Rather than directly filling in the RGB pixels in each frame, they treat video inpainting as a pixel propagation problem. But most of the methods produce a result that is often blurry or with strong artefacts. Moreover, both methods cannot handle high resolution videos with complex movements and the optimisation is quite time consuming.
That is why new approaches inheriting their principles and using deep learning have emerged. For instance, Wang [2] proposes a 3D encoder-decoder structure for video inpainting while Xu [9] exploits optical flow information to guide inpainting in videos. However, searching for correlated elements in a video is still not supported by these approaches. Of course, there are specialised methods for detecting, removing

or modifying specific types of effects, such as shadows [13], haze [15] and reflections, but they are not able to handle multiple effects.
Recently, a different view has yielded encouraging results. If our task is only to remove an object from a video, we can see the problem as a way to find layers, one of which contains our object and the other the background. Our problem can therefore be related to another area of video editing: image matting. Image matting is the process of accurately estimating the foreground object in images and videos [11]. This is a very important technique in film production for creating visual effects. It is different from image segmentation, because some pixels can belong to both foreground and background, these pixels are called partial or mixed pixels. This approach has seen many advances with neural networks. For instance, some suggest to use a context-aware natural image matting method for simultaneous foreground and alpha matte estimation [8].
Recently, a paper has found a way to automatically associate objects and their effects in a video. It is based on two methods: matting and flow-guided completion. The paper called *Omnimatte: Associating Objects and Their Effects in Video* [5] captures all the various scene effects that are correlated with the object and work with more than two layers. It is inspired by layered video decomposition and build on the work of Lu [4] that presents a method for decomposing a video into a set of human-specific RGBA layers, where each layer represents a person and their associated scene elements.

3. Problem definition

Image matting relies on a simple equation. Considering a frame I_t with pixels indexed by t seeing the pixels as a combination of foreground F_t and background B_t masks parametrized by the unknown matte estimation α_t, the classical matting equation is:

$$I_t = \alpha_t \cdot F_t + (1 - \alpha_t) \cdot \beta_t \ , \alpha_t \in [0;1]$$

It has seven unknown parameters (RGB for F_i, B_i plus α_i) and three known parameters (RGB for I_i). Here, our problem is slightly different and our goal is to decompose each frame into a set of color C_t and opacity α_t layers $\mathcal{L}_t$ for each objects present in the scene and indexed by i:

$$\mathcal{L}_t^i = \{C_t^i, \alpha_t^i\}$$

We will use a neural network to decompose the frame into layers. In addition to the frame I_t, it is reasonable to have, as input, a mask of the object we want to separate from the background M_t^i. Furthermore, in the case of videos, camera movements must be taken into account. We therefore introduce the estimated homography of the camera H_t. Then, as mentioned earlier, our model is based on the flow F_t. From the forward, backward flow is computed the confidence map W_t. Now our model can be written as the prediction of a color and opacity layer modeled by a neural network represented by the function f:

$$f(I_t, H_t, F_t, W_t, M_t^i) = \mathcal{L}_t^i = \{C_t^i, \alpha_t^i, \hat{F_t^i}\}$$

On the right term, we also want to predict the object flow $\hat{F_t^i}$. It will be useful during the optimisation. Once the problem has been posed, we can illustrate it as follows. On the left the frame F_t, the mask M_t, the flow F_t and a representation of the background Z_t. On the right, the alpha matte α_t, the reconstructed flow $\hat{F}_t$ and the colors $_t$.

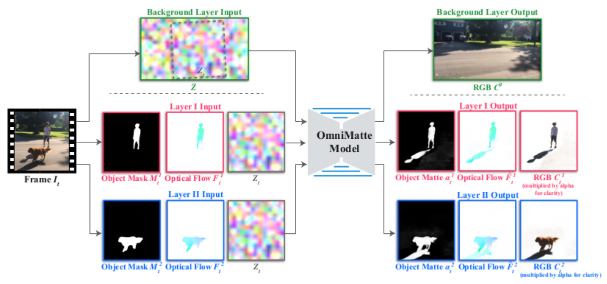


Figure 1: Illustration of the pipeline of a neural network able to solve the problem. ([5]

4. Methodology

4.1. Preprocessing

As detailed in the previous section, our network takes as input the homographies, the flow and its associated confidence map. Omnimatte repository does not give any solution to compute them so we have found solutions.

- Homographies: The original stabilisation code used in the paper to calculate homographies is currently internal to Google, so the authors have not been able to publish it. Therefore, three approaches were tested to calculate homographies between the first image and any other: SIFT, ORB and FLANN. As the official repository contains an example of homographies, we tested our algorithm to find a result close to the one they have. After these experiments, we decided to use SIFT by default. The algorithm consists mainly of five steps: peak selection in scale space, key point location, orientation assignment, key point descriptor and key point matching.

- Flow: To compute the forward and the backward flow, we have use RAFT [12]. RAFT is a recent deep network architecture for optical flow that extracts per pixel features, builds multi-scale 4D correlation volumes for all pairs of pixels, and iteratively updates a flow field through a recurrent unit that performs lookups on the correlation volumes.

- Confidence map: Once we have the two flows, confidence map can be calculated using the following formula:

$$W_t(p) = max(1 - e_t^{lr}(p), 0) \cdot \mathbb{1}_{e_p \leq \beta} \cdot M_t(p)$$

Where e_t^{lr} is the forward-backward flow error, β is an hyperparameter set to 20 and e_p measures the photometric error by : $e_p = ||Warp(I_{t+1}; F_{t,t+1}) - I_t||_1$. Intuitively, it looks only at objects and returns a weight taking into account the consistency of measurements between two flows and two RGB frames. The official repository provides the implementation.

- Optional: masks. When we download a video from YouTube, we do not have the masks of our objects, but we can use a detection algorithm like Cascade R-CNN network [16] available in the Detectron2 library. As the algorithm detects several objects in each frame, we decided to use a KD-Tree algorithm to automatically associate each box in the first frame with a box in the i-th frame. The algorithm calculates distances between the centres of the boxes and chooses the matching iteratively.

Even if there are many steps, the pre-processing time is negligible compared to the learning time.

4.2. Model

In order to solve the problem already presented, Omnimatte proposes a complex loss. To understand how the method works, it is important to understand each term of the loss.

- The main term is the reconstruction RGB loss. It forces the T layers to contain at each time all the information of a frame. It introduces the compositing order o_t that deals with overlapping objects. It should be seen as an order of superposition.

$$\mathcal{E}_{rgb} = \frac{1}{T} \sum_{t=1}^{T} ||I_t - Comp(\mathcal{L}_t, o_t)||_1$$

  Where $Comp$ refers to the composite, a way of taking into account information from the future.

- As there is a closed link between the input masks and mattes, they use a loss to coerce the opacity to match the input masks in order to guide the optimization in the early steps. This loss is turned off when its value is below a threshold set to 0.02.

$$\mathcal{E}_{mask} = \frac{1}{2 \cdot N \cdot T} \sum_{t,i} \frac{M_t^{i,+} \cdot ||M_t^i - \alpha_t^i||_1}{1 + \sum_{t,i} M_t^{i,+}} + \frac{M_t^{i,-} \cdot ||M_t^i - \alpha_t^i||_1}{1 + \sum_{t,i} M_t^{i,-}}$$

  Note: In the original article, they use boundary erosion but the code implements this loss which makes a difference between positive and negative mask values.

- Their experiments lead them to predict the flow. It serves as an auxiliary task that injects motion information into the model and improves the decomposition. It is a simple weighted reconstruction loss.

$$\mathcal{E}_{flow} = \frac{1}{T} \sum_{t=1}^{T} W_t \cdot ||F_t - Comp(\mathcal{F}_t, o_t)||_1$$

- The first regularisation term encourages the opacity to be sparse between each layers. It is a mix of L_1 and L_0 regularisation loss where $\phi_0(x) = 2\sigma(5x) - 1$ (σ is the sigmoid function). In the provided code, $\gamma = 0.01$ and $\phi_0 = 0.005$. Contrary to what is written in the paper, this loss is applied on the alpha composite.

$$\mathcal{E}_{\alpha-reg} = \frac{\gamma}{N.T} \sum_{t,i} ||Comp(\alpha_t^i, o_t)||_1 + \phi_0(Comp(\alpha_t^i, o_t))$$

- The second regularisation term and last loss term encourages temporal consistency within layers. Below, $Warp$ is a warped function.

$$\mathcal{E}_{\alpha-warp} = \frac{1}{N.T} \sum_{t,i} ||\alpha_t^i, Warp(\alpha_{t+1}^i, \mathcal{F}_t^i)||_1$$

  Finally the loss we need to optimize can be written as:

$$\begin{aligned} Loss = &\mathcal{E}_{rgb} + \lambda_m \mathcal{E}_{mask} + \mathcal{E}_{flow} \\ &+ \lambda_r \mathcal{E}_{\alpha-reg} + \lambda_w \mathcal{E}_{\alpha-warp} \end{aligned}$$

  As you can see, the optimisation problem is not easy to solve and I have not gone into detail so as not to make the approach too complex but the main idea to build RGBA layers is to optimise a reconstruction loss with additional regularisation terms taking into account the composition order and the composite.

4.3. Limitations

Pre-processing:
In our experiments, we found that calculating homographies is not always an easy task and that it greatly influences the results. We therefore always examined the results to see if they were consistent.
The pre-processing steps are complicated but not time consuming compare to the training time. The network is slow. The paper states their algorithm runs 2 hours on two P100 with a batch of 16. With google colab pro and a single P100, it takes around 6 hours with a limited batch size of 8 to train a 480 pixels video.

Loss:
The $\alpha - reg$ loss aims to prevent a trivial solution where one layer reconstructs the entire image but can also enable an object to be reconstructed in multiple object layers. Singular situations may arise during optimisation.

Optimisation:
It is not trivial to know when to stop the algorithm if we don't look at the intermediate results. The official repository trains until a certain number of iterations, but the number depends on the batch size and is video dependent. To have a fair comparison, we have trained the network for at most 2.000 epochs event if the training process recommends to stop after.

5. Evaluation

5.1. Dataset

In a first step, we used the 480p DAVIS dataset [6] to familiarise ourselves with the model and try to reproduce the results. It contains short annotated videos of 50-70 frames with many environments, effects and movements. The videos are not synthetic. Then we worked on videos chosen from YouTube without annotation.

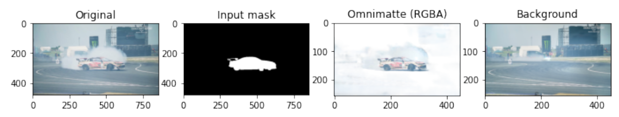
5.2. Metrics

For image painting, quantitative metrics are used because we can easily have the ground truth. For video inpainting, the ground truth is nearly never given and impossible to have in a dynamic world. We have to distinguish two cases. The first one is a video with static camera. Under certain conditions, we can choose a frame representing the entire background as a reference and compare the metrics between the frames. The second one is a video with camera movement. In this case, the ground truth is intractable so we can not compare methods easily. In the literature, the visual rendering often serves as a comparison.

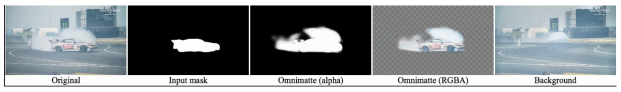
5.3. Experiments

5.3.1 Reproducing the results of the paper

First, we tried to reproduce the results of the paper using the "blackswan" and "drift-chicane" videos from the DAVIS dataset. They contain about 50-70 images with a water or dust motion effect. In both videos, the object covers a significant part of the image.



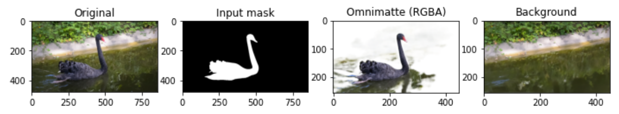
(a) Our results



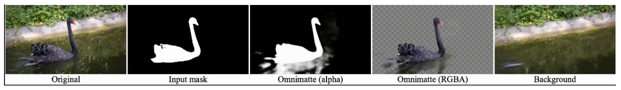
(b) Results from the paper

Figure 2: Comparison between the paper and our experiments on the video "drift-chicane"

Impressively, our network was able to correlate the dust with the car and is able to handle variable size with a large correlated effect. The drift-chicane is a textbook case since the camera is static and there is only one correlation. But it is really interesting because the shape of the car varies over time and the dust is very prominent in the images.



(a) Our results



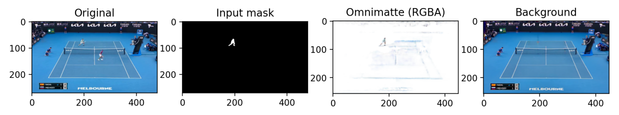
(b) Results from the paper

Figure 3: Comparison between the paper and our experiments on the video "blackswan"

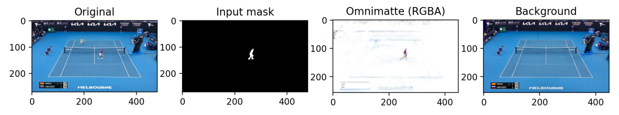
Blackswan has other interesting points. Starting with the movement of the water and the reflection of the swan on the water, which correlates with the swan. You can even make out the shape of the neck in the reflection of the swan on the water. Furthermore, the dynamic background adds a layer of complexity. Here homographies play an important role in the result.

5.3.2 Using a video from internet

Next, we wanted to see if the network could work on any video available on the Internet. To do this, we automatically downloaded a video from the Internet, extracted the frames, downsampled the frames to 60 frames, created masks and followed the pipeline. The video is an excerpt from a set of the tennis match between Nadal and Medvedev at the last Australian Grand Slam tournament. Unlike previous experiments, we used several masks: one for Nadal, the other for Medvedev. The network therefore has to handle two masks at different scales since Medvedev is always further away from the camera than Nadal.



(a) First layer



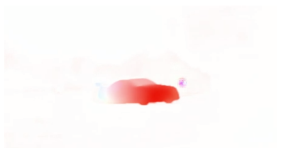
(b) Secondlayer

Figure 4: Results on a video taken from internet

Omnimatte removes both masks in one training session. It correctly correlates players with their small shadows cast on the floor but slightly correlates players with the scoring lines. The results are satisfactory, especially as the tennis ball that was not labelled was removed. It automatically understood that the ball was correlated to the players and therefore deleted it.

5.3.3 Flow

As explained, our network estimates the flow to innject motion information into the model. On ”drift-chicane”, if we take a look at the computed flow, we see that it is well reconstructed. However, the network was not able to catch the dynamic of the background screen. Overall, the flow is the same, but I show a photo illustrating their difference below.


(a) Our reconstruction

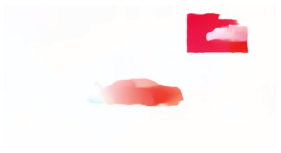
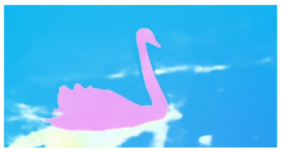

(b) Calculated by RAFT

Figure 5: Comparison of flows for drift-chicane

The reconstruction for the blackswan captures the swan’s reflection but has difficulty attributing the same speed to the swan and its reflection, whereas RAFT succeeds.


(a) Our reconstruction

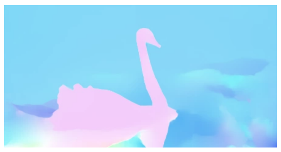
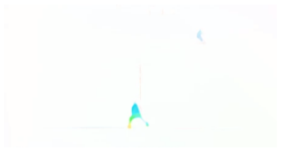

(b) Calculated by RAFT

Figure 6: Comparison of flows for ”blackswan”

For the ”Nadal vs Medvedev” video, even if the overall rendering is visually similar, Medvedev’s flow is sometimes miscalculated with the reconstruction and we can’t recognize his body while with RAFT, the shapes are more marked and the colours more visible.


(a) Our reconstruction

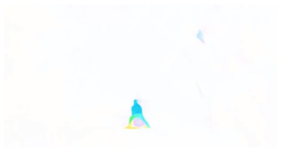

(b) Calculated by RAFT

Figure 7: Comparison of flows for ”Nadal vs Medvedev”

5.3.4 Comparison with FGVC

In order to criticise our network, we have compared it with FGVC [7]. It belongs to the family of flow-guided networks and aims to solve a problem similar to the one presented above. Their network is able to remove objects from a video as they show on their website [1] but it uses masks including effects. So, to have a fair comparison, we fed the network with the masks used in Omnimatte.

[1]https://filebox.ece.vt.edu/~chengao/FGVC/pages/object_removal.html

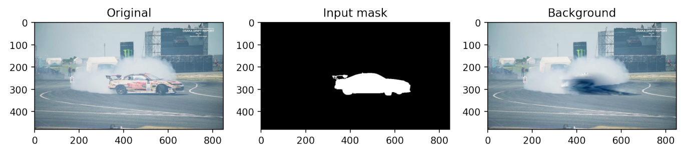


Figure 8: FGVC result for ”drift-chicane”

The dust effect is not handle by FGVC. The result is unsatisfactory while it looks great with Omnimatte. On the video, the car is well removed but the rendering is not realistic and a spot appears. Since the video is static, we can compare quantitatively the two approaches. In both case, we use the first rgb frame filtered by the input mask as the groundtruth.

Metrics (mean)	Omnimatte	FGVC
PSNR	41.8	33.2
SSIM	0.98	0.91

Table 1: Quantitative comparison between Omnimatte and FGVC

Omnimatte definitely gives the best results. Since we are using the first image as a reference, it is normal not to have a perfect PSNR because some elements are moving in the video, like the screen at the back of the image or the young women.

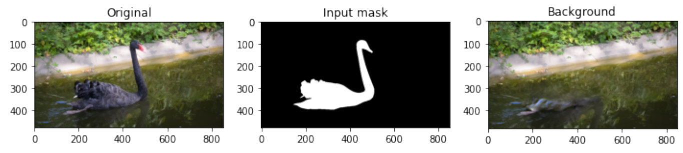


Figure 9: FGVC result for ”blackswan”

For the blackswan, given our mask, FGVC is unable to remove the reflection of the swan and the water movement but the swan is indeed removed. The result does not look real.

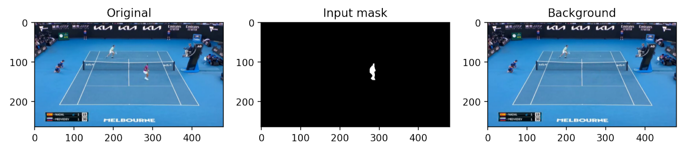


Figure 10: FGVC result for ”Nadal vs Medvedev”

Finally, the video ”Nadal vs Medvedev” shows us that the network is not as good as Omnimatte at handling fast movements, as you can see the blur of Nadal’s form. Moreover, the network is not able to remove several objects at the same time and did not remove the tennis ball. We have to iterate the training pipeline several times in order to perform multiple removals. About metrics, Omnimatte returns again a better PSNR and SSIM which proves that the network is better and time coherent.

Metrics (mean)	Omnimatte	FGVC
PSNR	40.6	34.3
SSIM	0.98	0.97

Table 2: Quantitative comparison between Omnimatte and FGVC

6. Conclusion

Omnimatte seems to be a promising network for object removal. It is able to automatically remove multiple objects and their correlations in videos using a matting approach. Moreover, once the layers have been calculated for each object, a wide variety of transformations can be accessed: multiplication of objects, freezing of a moving object, color pop, etc.
However, the training time is long compared to other networks. Once we have downloaded the weight of the pre-trained model, using FGVC on a single video of 50-70 frames can be processed in several minutes, 10 is an upper limit whereas Omnimatte takes several hours. During our experiences, at 2 hours we begin to have good results, after 5 hours our networks has converged. The time scale is not comparable. In addition, Omnimatte is limited in memory whereas FGVC can take larger images as input. Thus, depending on the intended use, one network will be more suitable than another. Omnimatte is encouraging but cannot be used yet on a common software.
Still, the work done on this subject is very promising; there is a great effort to continue to understand how the optimisation works in order to speed it up and to have a better understanding of the convergence. As for the method itself, it cannot separate objects or effects that remain completely stationary with respect to the background throughout the video. However, constructing a representation of the background that explicitly models the 3D structure of the scene may be one way to solve these problems.

7. Supplementary Materials:

All the presented results are available on the following link. [2] We strongly recommend you to watch the videos.

References

[1] Patwardhan et al. "Video inpainting of occluding and occluded object". In: 2005.

[2] H. Huang et al. C. Wang. "Video inpainting by jointly learning temporal structure and spatial details." In: *AAAI*. 2019.

[3] P.Krahenbuhl et al. D. Pathak. "Context Encoders: Feature Learning by Inpainting". In: *CVPR*. 2016.

[4] F. Cole et al. E. Lu. "Layered neural rendering for retiming people in video." In: *SIGGRAPH*. 2020.

[5] C. Forrester et al. E.Lu. "Omnimatte: Associating Objects and Their Effects in Video". In: *CVPR*. 2021.

[6] J.Pont-Tuset et al. F.Perazzi. "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation". In: *CVPR*. 2016.

[7] Saraf et al. Gao. "Flow-edge Guided Video Completion". In: *European Conference on Computer Vision*. 2020.

[8] Q. Hou and F.Liu. "Context-Aware Image Matting for Simultaneous Foreground and Alpha Estimation". In: *ICCV*. 2019.

[9] X. Li et al. R. Xu. "Deep flow-guided video inpainting." In: *CVPR*. 2019.

[10] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. "SinGAN: Learning a Generative Model from a Single Natural Image". In: *Computer Vision (ICCV), IEEE International Conference on*. 2019.

[11] V.Jayaram et al. S.Sengupta. "Background Matting: The World is Your Green Screen." In: *CVPR*. 2020.

[12] Z. Teed and J. Deng. "RAFT: Recurrent All Pairs Field Transforms for Optical Flow". In: *ECCV*. 2020.

[13] Tianyu Wang et al. "Instance Shadow Detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[14] Shechtman Wexler and Irani. "Space-Time Completion of Video". In: *IEEE*. 2007.

[15] Y.Bahat and M.Irani. "Blind Dehazing Using Internal Patch Recurrence". In: *ICCP*. 2016.

[16] N. Vasconcelos Z. Cai. "Cascade R-CNN: Delving into High Quality Object Detection". In: 2017.

[2] https://drive.google.com/drive/folders/1jK7FtTWspb2B5IDt56uPST2KxRi42amj?usp=sharing