Interactive video stylization using few-shot patch-based training

Prof.	☐
코드 깃허브 링크	
Reviewed	☐
작성일시	@March 10, 2022 5:02 PM
Paper (해당시)	https://arxiv.org/abs/2004.14489
최종 편집	@October 7, 2022 1:28 AM
Note (추가info)	
Reviewer (담당자)	㉾ 김하연
발표영상 (해당시)	
Keywords	Paper review / Study Style Transfer
추가 자료	
PPT (해당시)	
발표일자	@March 25, 2022

Summary

1. Few-shot patch-based training strategy
2. Parallel processing, random access, real-time inference(live video stream)

Abstract

> **한줄소개 및 장점 :** In this paper, we present a learning-based method to the **keyframe-based video stylization** that allows an artist to propagate the style from a few selected keyframes to the rest of the sequence. Its key advantage is that the resulting stylization is semantically meaningful, i.e., specific parts of moving objects are stylized according to the artist’s intention.

> **기존 연구와의 비교 :** In contrast to previous style transfer techniques, our approach does not require any lengthy pre-training process nor a large training dataset. We demonstrate how to train an appearance translation network from scratch using only a few stylized exemplars while implicitly preserving temporal consistency.

> **결과 1 :** This leads to a video stylization framework that supports real-time inference, parallel processing, and random access to an arbitrary output frame. It can also merge the content from multiple keyframes without the need to perform an explicit blending operation.

> **결과 2 :** We demonstrate its practical utility in various interactive scenarios, where the user paints over a selected keyframe and sees her style transferred to an existing recorded sequence or a live video stream.

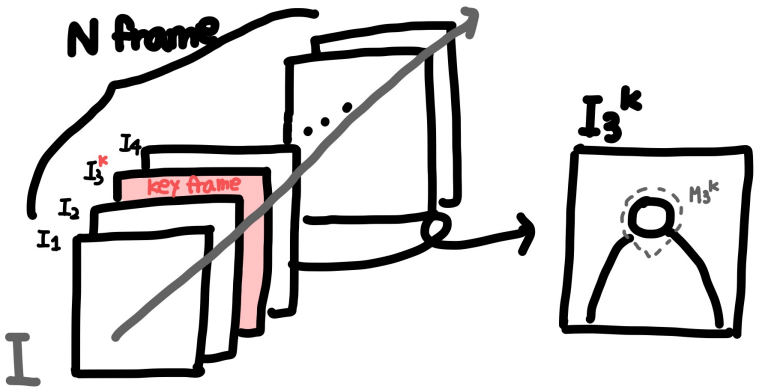
소수의 stylized exemplar와 keyframe을 이용하여 실시간 inference, 병렬 처리, 랜덤 접근을 가능하도록 한다.

Introduction

- Example-based stylization : adding specific style-aware content loss, control이 힘듦 (statistical correlation만 측정하기 때문에)

- Keyframe-based video stylization : employ patch-based synthesis(semantically meaningful transfer), full control 가능, but 생산 환경에 따라 여전히 해결해야 할 문제 존재
  - Sequential fashion : random access, parallel processing 불가능
  - Merging or blending : 2개 이상의 keyframe으로 부터 스타일화 되어진 content는 visible clutter or ghosting artifacts 존재 → 이를 해결하려면 어렵고 힘겹게 해결 가능(하나의 keyframe을 먼저 고르고 이후 previous synthesis run의 결과에 색칠함으로써 corrective keyframe을 준비)
- 목표 : semantically meaningful transfer, maintaining temporal coherence, react adaptively to incoming user edits, integrate them on the fly
- 이 논문의 제안
  - Appearance translation framework for arbitrary video sequence : semantically meaningful style transfer without any lengthy domain-specific pre-training(like few-shot)
  - Patch-based training mechanism : i2i translation network to generalize in a setting
  - 결과 : parallel / live video stream stylize 가능(productive workflow)

Our Approach



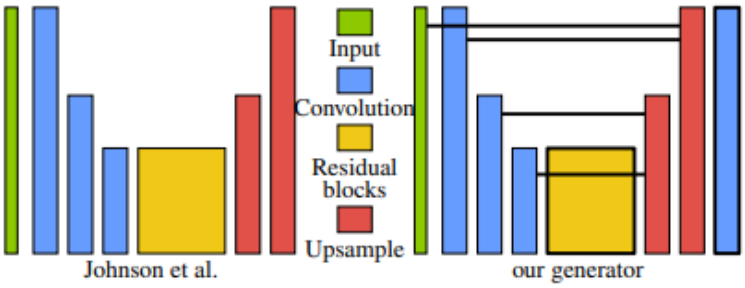
input : video sequence I which consists of N frames, output : sequence O

mask : every frame I_i는 mask M_i를 가질 수 있음(관심 있는 영역만 style이 입히도록)

keyframes : $I^k \subset I$, stylized keyframe S^k → keyframe 전체 / 일정 영역(M^k)만 stylize

- our task : S^k의 style로부터 전체 I를 consistent하게 stylize ⇒ visual quality, temporal consistency + random order, in-parallel or on-demand in real-time(이전 frame이 style이 되거나 merging이 되는 것을 기다릴 필요X, 실시간)
- Translation filter : heterogeneously hand-drawn exemplars S^k로 부터 스타일을 배워서 I를 병렬적으로 / 필요한 하나의 frame에 stylize
  - U-net-based image-to-image translation framework[Futschik et al. 2019] 이용

    ▼ Futschik et al.



U-Net은 이미지 분할을 목적으로 제안된 End to End 방식의 Fully Convolutional Network 기반 모델이다.(U처럼 생겨서 u-net이라고 부른다)Futschik 논문은 Johnson et al.의 방식을 베이스 모델로 하는데 아래와 같은 이유로 convolution filter의 사이즈를 변경하고, convolutional layer를 추가하며 skip connection을 만들어 냈다.

> we changed the size of convolutional filters in the very first layer from 9 × 9 to 7 × 7 and in the very last layer of the original architecture from 9 × 9 to 5 × 5. We increased the number of residual blocks used from five to nine. Next, we added **skip connections** using concatenation of feature maps [RFB15] to the upsampling layers, which has been shown to improve gradient propagation, and we replace convolutions with fractional strides with nearest neighbor upsampling followed by an additional 3×3

> convolution. Lastly, we attached two more convolutional layers before the output, which we observed have positive effect when the skip connections are added.

  - high frequency detail을 얻는 custom network architecture
  - 기존 Futschik의 network는 큰 데이터셋으로 학습하여, 우리 방식(적은 샘플)에 바로 적용하기에는 strong overfitting 발생, keyframe은 완벽하게 reconstruct 가능하나, 다른 frame이 좋지 못하게 stylize되어진다.

- 위 문제를 해결하기 위해 수정 : network의 훈련 방법(**patch-based training strategy**), 최적화 문제 formulate(fine-tuning, **hyper-parameter optimization**)
- 추가적으로, consistency를 측정할 필요 없이 temporal flicker를 해결(**temporal coherency**)

1. Patch-based training strategy

: S^k로 부터 작은 사각형 패치를 랜덤하게 샘플링(M^k내의 영역에서만), 같은 사이즈의 사각형 패치 영역을 predict하면서 network를 훈련시킨다.

- Note that thanks to the fully convolutional nature of the network, once trained, it can be directly used to stylize the entire video frame even though the training was performed on smaller patches

▼ Fig. 5

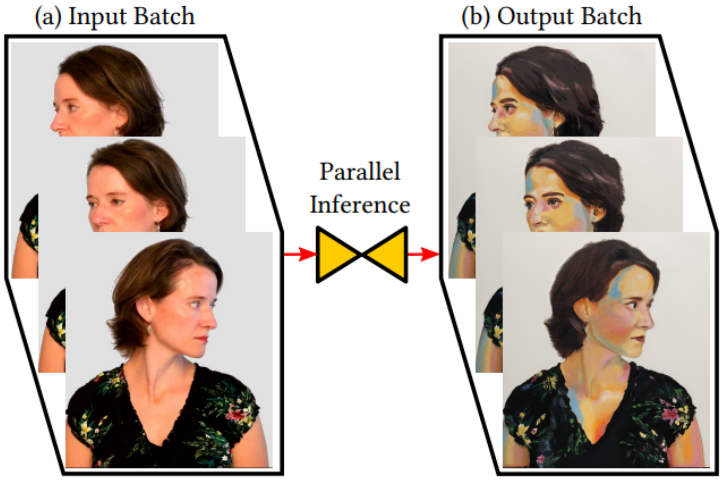


Fig. 5. Inference: thanks to the fully convolutional nature of the network, we can perform the inference on entire video frames, even though the training is done on small patches only. Since the inference does not depend on other stylized frames, all video frames can be stylized in parallel or in random order. This allows us to pass many or even all of the input frames (a) through the network in a single batch and get all output frames (b) at once. Video frames (left) courtesy of © Zuzana Studená.

- cropping & randomization step : 크고 다양한 dataset이 들어온 것 처럼 stimulate(overfitting 방지, generalize to stylize)
- reconstruction loss : S^k에서 full-frame training(Futscik)보다 reconstruction loss ↑, but remaining frame에서 reconstruction loss ↓(이게 훨씬 visual quality가 좋음)

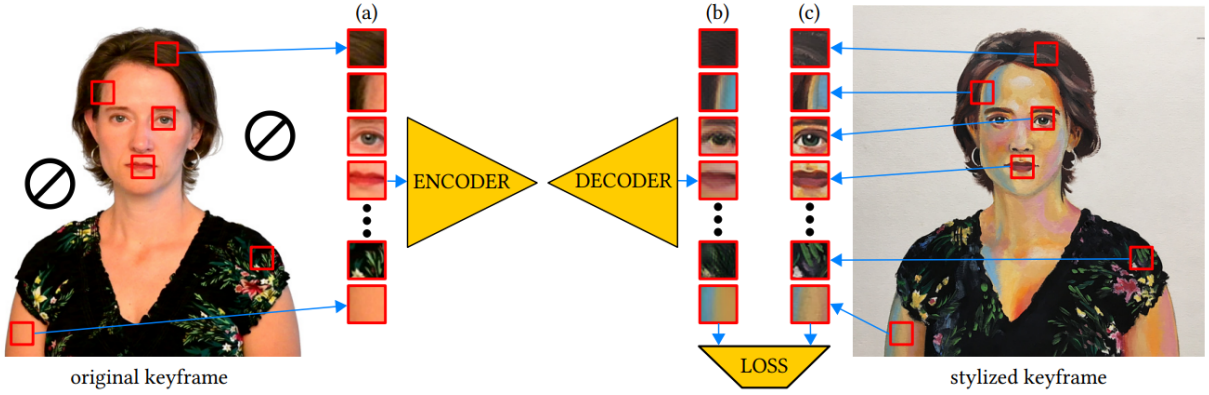


Fig. 4. Training strategy: we randomly sample a set of small patches from the masked area of the original keyframe (a). These patches are then propagated through the network in a single batch to produce their stylized counterparts (b). We then compute the loss of these stylized counterparts (b) with respect to the co-located patches sampled from the stylized keyframe (c) and back-propagate the error. Such a training scheme is not limited to any particular loss function; in this paper, we use a combination of L1 loss, adversarial loss, and VGG loss as described in [Futschik et al. 2019]. Video frame (left) and style exemplar (right) courtesy of © Zuzana Studená.

2. Hyper-parameter Optimization

: overfitting 문제를 patch-based training으로 충분히 해결이 되지만, hyper-parameter를 적절하게 세팅하는 것도 중요하다는 것을 발견, loss function을 minimize하는 방향으로 !

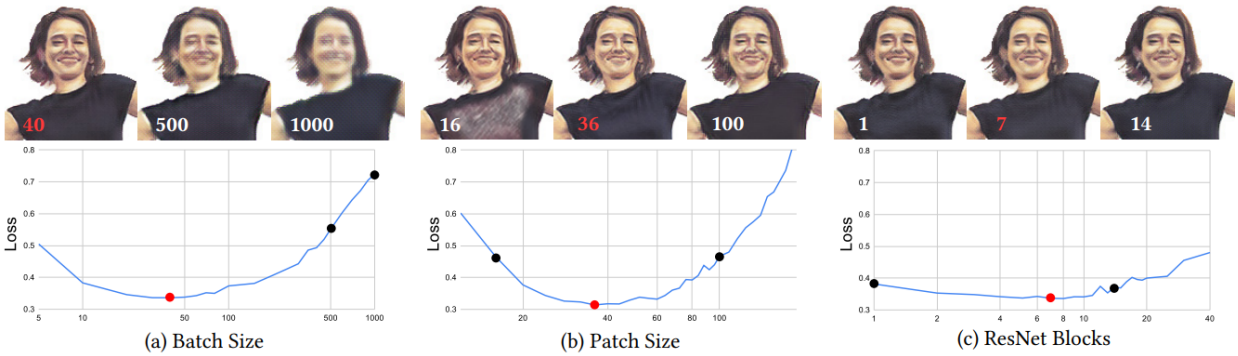


Fig. 8. Influence of important hyper-parameters on visual quality of results. The loss, y-axes, is computed w.r.t. the output of Jamriška et al. [2019]. The best setting for each hyper-parameter is highlighted in red: (a) The loss curve for the batch size N_b—the number of patches in one training batch (other hyper-parameters are fixed). As can be seen, increasing N_b deteriorates visual quality significantly; it indicates that there exists an ideal amount of data to pass through the network during the back-propagation step. (b) The loss curve for the patch size W_p. The optimal size of a patch is around 36x36 pixels. This fact indicates that smaller patches may not provide sufficient context while larger ones could make the network less robust to deformation changes. (c) The loss curve for the number of ResNet blocks N_r that corresponds to the capacity of the network. As can be seen, settings with 7 ResNet blocks is slightly better than other results; however, this hyper-parameter does have major impact on the quality of results. For additional experiments with hyper-parameter setting, refer to our supplementary text.

1) W_p : training patch size

2) N_b : patch들의 수

3) N_r : ResNet block의 수

4) constraint : T_t(network가 train하는 시간), T_i(한개의 비디오 프레임의 inference 시간) ⇒ 이 논문에서 T_t=30s, T_i=0.06s이므로, full optimization of hyper-parameters는 추적할 만 하다

5) **grid search** : optimal value 찾기, 다른 validation sequences가 사용되어도 consistent 함.

▼ Grid search, Fig. 6

1. **Grid search**란? 모델 하이퍼 파라미터에 넣을 수 있는 값들을 순차적으로 입력한뒤에 가장 높은 성능을 보이는 하이퍼 파라미터들을 찾는 탐색 방법
2. Fig. 6

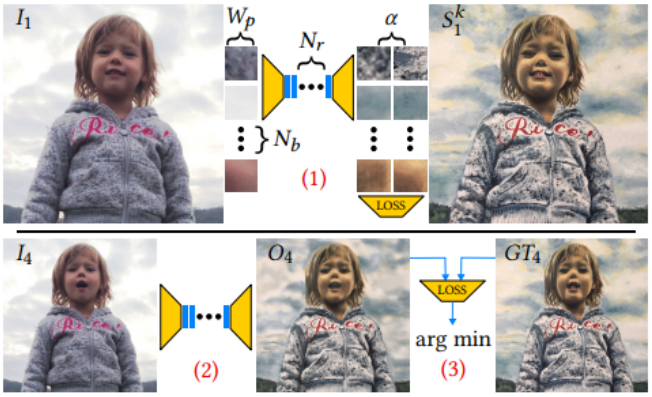


Fig. 6. To fine-tune critical hyper-parameters of our network, we propose the following optimization scheme. We tune batch size N_b, patch size W_p, number of ResNet blocks N_r, and learning rate α. Using the grid search method we sample 4-dimensional space given by these hyper-parameters and for every hyper-parameter setting we (1) perform a training for a given amount of time, (2) do inference on unseen frames, and (3) compute the loss between inferred frames (O_4) and result of [Jamriška et al. 2019] (GT_4) - which we consider to be ground truth. The objective is to minimize this loss. Note that the loss in step (1) and the loss in step (3) are both the same. Video frames (I) and style exemplar (S) courtesy of © Zuzana Studená.

3. Temporal Coherency

: frame-independent inference를 수행하면서 temporal flicker(temporal noise, visual ambiguity of stylized content 때문)를 억제하는게 목표

- appearance translation network : amplify temporal noise(input video에서 작은 temporal instability도 큰 visible flicker 발생) ⇒ 시간의 영역에서 작동하는 **motion-compensated variant of bilateral filter** 사용
  - bilateral filter의 경우 nearby frame을 require → but, frame-independent processing을 방해하는건 x
  - distinct and variable texture를 가진 물체에만 temporal flicker 감소(visual ambiguity 때문에discriminatory information이 부족한 부분은 flicker) ⇒ distinctive region을 가진 scene만 가져오는건 실용적이지 못하기 때문에, network에 **추가적인 input layer**(2D gaussian의 sparse set → unique color variation을 제공)를 제공 : discriminative power ↑
- Gaussian = points attached to a grid(random colorization → better localization & sparsity 제공, local distortion ↓
  - block-matching : 그리드 위 optimal translation of each point 계산
  - ARAP deformation : grid 구조를 regularize

▼ Fig. 7

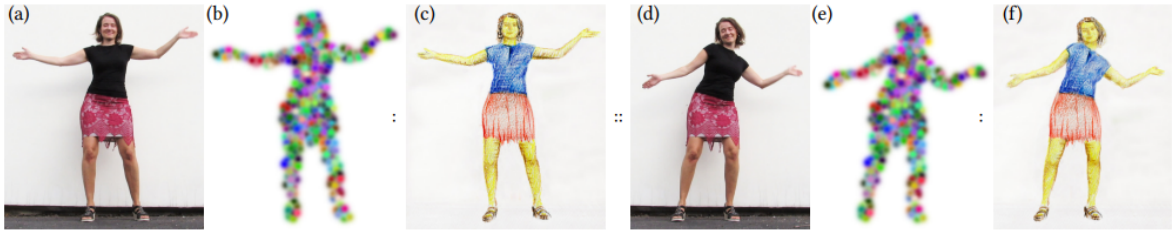


Fig. 7. To suppress visual ambiguity of the dark mostly homogeneous T-shirt in (a) an auxiliary input layer is provided that contains a mixture of randomly distributed and colored Gaussians (b). The translation network is trained on patches of which input pixels contain those additional color components. The aim is to reproduce the stylized counterpart (c). Once the network is trained a different frame from the sequence can be stylized (d) using adopted version of the auxiliary input layer (e). The resulting sequence of stylized frames (f) has notably better temporal stability (cf. our supplementary video at 2:40). Video frames (a, d) courtesy of © Zuzana Studená and style exemplar (b) courtesy of © Pavla Sýkorová.

Results

- overfitting을 방지하기 위해서 full-frame training에 아무리 여러 data augmentation을 시행해도 이 논문의 patch-based training이 더 뛰어난 성능을 보여줌.

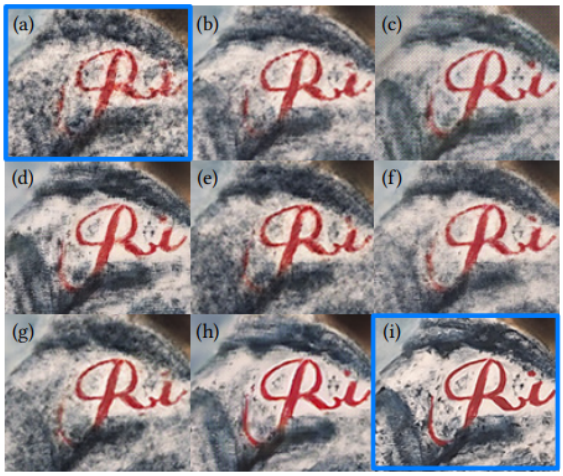


Fig. 9. To deal with the overfitting caused by a minimal amount of training data, we tried several commonly used techniques to enforce regularization. In all cases shown in this figure, we trained the network on the first frame; the shown results are zoomed details of the fifth frame. (a) is a result of the original full-frame training. (b-h) are results of full-frame training with some data augmentation. (i) is a result of our patch-based training strategy—see how our technique can deliver much sharper and significantly better visual quality results, please, zoom into the figure to better appreciate the difference. In case of (b-c), Gaussian noise was used to augment the data; (d) some pixels were randomly set to black; (e-f) some parts of the image were occluded; (g) dropout of entire 2D feature maps; (h) dropout of individual pixels before each convolution layer.

- SOTA(Jamriska et al.)와 quality 비교 :
  - SOTA의 경우, non-parametric 모델로 target frame의 style bitmap의 큰 chunk를 복제하므로 origina style exemplar의 texture를 잘 보존한다.
  - 반면, 이 논문의 경우 parametric 모델로 target frame의 구조적 디테일을 잘 잡아낸다.
- SOTA와 temporal consistency 비교 : multiple keyframe을 사용한 경우 ghosting artifact가 SOTA보다 현저히 줄어들었다. 하지만, original noisy sequen를 사용할 때에는 조금 더 flicker 현상이 나타났는데, 이는 우리의 방식은 random access & parallel processing이 가능하므로 충분히 질적으로 향상했다고 평가한다.
- Live stream : 가장 큰 장점 중 하나로 live video stream을 실시간으로 style transfer가 적용된 모델을 볼 수 있다는 것이다.
- Interactive application : real-time feedback을 받는 것과 artifact를 줄인 것이 큰 이점으로 나타났다.

Conclusion

Limitations and future work

한계 : 물체가 회전해서 보이지 않던 content를 보여주는 경우 stylize가 힘듦 → 본래 keyframe에서 stylize되지 않았던 새로운 feature를 consistent하게 stylize를 하는 것은 힘듦으로, additional keyframe 필요(consistent를 위해)

고해상도(eg. 4K) stylize 어려움 → 훈련에 시간 많이 소요

앞으로 연구 : implement real-time variants of the motion-compensated bilateral filter + mixture of colored Gaussians

Conclusion

1. Few-shot learning : 한 개 / 적은 개수의 손으로 그린 keyframe을 이용하여 전체 target sequence를 stylize
2. Patch-based training, optimized hyper-parameter, temporal coherence : translation network가 적은 샘플로 학습 가능 + temporally coherent stylized video

3. Frame-independent mode : random access & parallel processing이 가능한 전문가 비디오 편집에 용이해짐

추가 정리

Q. **Real time inference time때도 patch-based로 진행하는가 ?**

: https://www.youtube.com/watch?v=DDlYIzhoXfI&t=2019s 위 영상 16:25초를 참고하면, 저자는 patch-based로 학습을 하게되면 엄청 느리게 하기 때문에 patch로는 어렵다고 말하였습니다. network가 fully convolutional layer기 때문에 전체 이미지를 넣어서 네트워크에 통과시킬 수 있다고 말합니다.(아니, 그러면 이 논문을 쓴 게 큰 의미가 없지 않나..?)

- convolution 연산에서 patch 대신 연산 해줄만 한 것이 존재 or 전체 이미지가 convolution 연산에 들어간 것..

Q. **mask는 어떻게 stylize되는 것인가?**

: https://www.youtube.com/watch?v=NwhgHZQvsYI&t=2s 위 영상 11:44초 ~ 12:13초, 14:30초를 참고하면, mask는 새로운 레이어를 하나 깔기 보다는, 색칠된 영역(stylized region) 자체를 의미하는 것 같습니다. → 즉, 이전 논문에서는 blending(두 개 이상의 style을 합치는 것)을 하는 경우, 앞의 것이 끝나면 추가로 또 색칠하면서(즉, frame 전체를 stylize를 하기 위해서는 계속 모든 frame 마다 그 전체를 색칠했어야 합니다), sequential하게 진행 되었지만, 이 논문에서는 patch-based learning으로 parallel processing이 가능해지면서, 굳이 모든 프레임의 전체를 색칠할 필요 없이, 얼굴이면 얼굴같이 전체가 아닌 부분적으로 칠하여도 style transfer가 잘 진행되는 것을 확인할 수 있습니다. 이처럼 얼굴만 따로 색칠 / 몸만 따로 색칠 처럼 그 색칠한 영역을 mask를 의미하는 것 같습니다.(전체를 색칠했다면 마스크는 전체 영역이 되겠죠?)

Q. mask 외부 영역은 제외하고 patch를 추출한다고 했는데 어떻게 되는 것인가?

: 본 논문을 참고하면, 아래와 같이 적혀있으며, Fig.4의 왼쪽 사진처럼 mask 외부 지역에 X표시가 되어있는걸 확인할 수 있습니다. 현재 확인한 코드 / 논문 상에서 mask 외부 영역을 제외하는 방법(foreground, background 떼어내는 방법 등)은 없었습니다. 위에 질문의 답변과 연결하여 생각해보자면, mask가 일종의 색칠한 영역이라면 샘플링은 색칠된 영역 내에서만 진행된다는 것이니, 랜덤으로 샘플링 한 patch 중 mask 영역에 해당되지 않으면 제거한다는 의미가 아닐까요..!?(사실, 이건 많이 헷갈리는 부분이라 더 알아봐야할 것 같습니다!)

> The sampling is performed only within the area of masked pixels M^k.

← 생각의 근원 : https://www.youtube.com/watch?v=NwhgHZQvsYI&t=2s 위 영상 4:21초 쯤 보면 patch는 사람이 아닌 부분도 랜덤하게 샘플링 된 것을 볼 수 있습니다. 그래서, 전경과 후경을 따로 떼어내는 알고리즘이 있다기 보단, 랜덤하게 샘플링 한 후 mask 영역 내에 존재하는 patch만 남긴 것이 아닐까 라는 생각이 들었습니다!

References

1. SIGRAPH 2020 Real-Time Live : https://www.youtube.com/watch?v=DDlYIzhoXfI&t=2019s
2. SIGRAPH 2020 Talk : https://www.youtube.com/watch?v=NwhgHZQvsYI&t=2s