

# Supplementary Material: Mitigating Reverse Engineering Attacks on Local Feature Descriptors

Deeksha Dangwal<sup>1</sup>  
deeksha@cs.ucsb.edu

Vincent T. Lee<sup>2</sup>  
vtlee@fb.com

Hyo Jin Kim<sup>2</sup>  
hyojinkim@fb.com

Tianwei Shen<sup>2</sup>  
tianweishen@fb.com

Meghan Cowan<sup>2</sup>  
meghancowan@fb.com

Rajvi Shah<sup>2</sup>  
rajvishah@fb.com

Caroline Trippel<sup>3</sup>  
trippel@stanford.edu

Brandon Reagen<sup>4</sup>  
bjr5@nyu.edu

Timothy Sherwood<sup>1</sup>  
sherwood@cs.ucsb.edu

Vasileios Balntas<sup>2</sup>  
vassileios@fb.com

Armin Alaghi<sup>2</sup>  
alaghi@fb.com

Eddy Ilg<sup>2</sup>  
eddyilg@fb.com

<sup>1</sup> University of California, Santa Barbara  
Santa Barbara, USA

<sup>2</sup> Facebook Reality Labs Research  
Redmond, USA

<sup>3</sup> Stanford University  
Stanford, USA

<sup>4</sup> New York University  
New York, USA

---

## 1 Comparison to Prior Work

We compare our work against several prior works that attempt to reverse engineer RGB images from features. **Figure 1** compares our reverse-engineered image results compared to that of d’Angelo et al. [3] and Weinzaepfel et al. [2]. Compared to the latter in **Figure 1(e)**, our result using SIFT shown in **Figure 1(b)** produces a qualitatively better reverse-engineered image with more accurate color estimates. As shown in **Figure 1(f)**, the work from d’Angelo

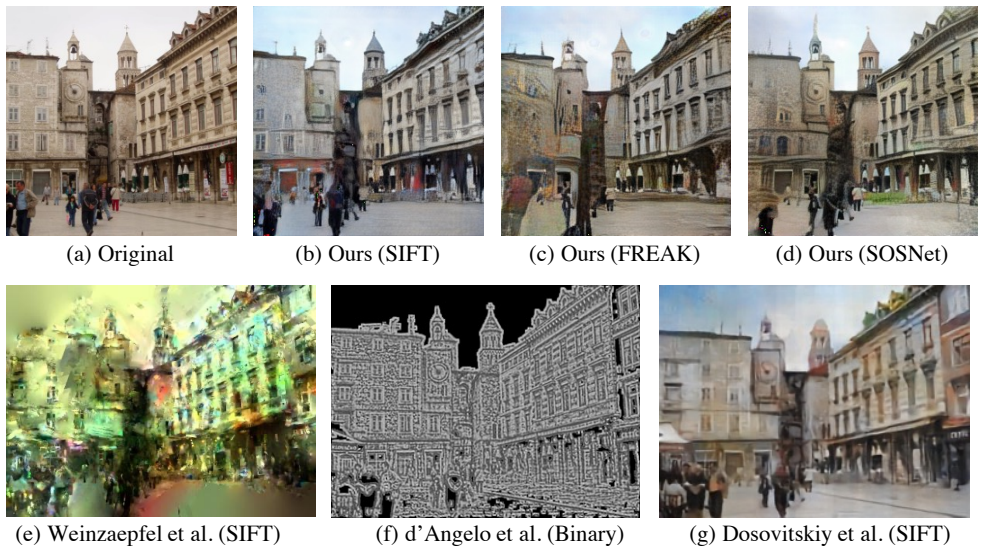


Figure 1: Top from left to right. Ground truth image and our reconstructions from SIFT, FREAK, and SOSNet sparse feature maps. Bottom from left to right. Reconstructions by prior work from binarized descriptors in [3], and SIFT features in [2].

et al. reconstructs image gradients only and is not comparable to our work. We also compare our results to those by Dosovitskiy and Brox [5] in Figure 1(g). In contrast to our work, Dosovitskiy and Brox use more keypoints and descriptors for their reconstruction using SIFT descriptors; they use roughly 3000 keypoints to reconstruct this image while we use 1,000 or fewer in our experiments. Qualitatively the results are comparable.

The previous state of the art is recent work proposed by Pittaluga et al. [6] which also uses convolutional neural networks to reverse-engineer images. Pittaluga et al. use additional information such as depth and RGB at the keypoint location to supplement SIFT descriptors as input to their reverse engineering model. Our work does not use depth nor RGB information, and does not make use of a separate network for visibility estimation (as the VisibNet from [7]). We also compare against FREAK and SOSNet descriptors while Pittaluga et al. exclusively analyze SIFT descriptors.

The results show that even without the additional depth and RGB information from Pittaluga et al., our reconstructions produce more detail and more accurate color in average in the cases of SIFT and SOSNet. In contrast, FREAK does not allow us to reconstruct the color information as well and we see some color artifacts (e.g., see the clock image). Since a practical reverse engineering attack for a relocalization service does not provide depth or RGB information to the honest-but-curious adversary, our attack formulation aligns with the real-world scenario. When using all input data assets (depth, SIFT, and RGB) Pittaluga et al. achieve a maximum average SSIM of 0.631 on reconstructions and an average SSIM of 0.578 when using only SIFT descriptors (Table 1). In contrast, our reverse engineering attack yields an average SSIM of 0.675 for reconstructions from SIFT features alone and thus provides a new state of the art. We attribute the improvements to our architecture choice and training procedure which we describe below.

	Inputs	SSIM
Prior Work [7]	Depth Only	0.578
	Depth+SIFT	0.597
	Depth+SIFT+RGB	0.631
Ours	SIFT Only	0.675
	FREAK Only	0.511
	SOSNet Only	0.616

Table 1: Comparison of average SSIM values of the reverse engineered images from prior work [7] and our work. Our work achieves better SSIM results for SIFT without using inputs like depth or RGB.

## 2 Architecture Implementation Details

Our reverse engineering attack uses a deep convolutional generator-discriminator network (see main paper). We provide the implementation details of our reverse engineering network, including architecture, optimization, and training methodology in this section.

### 2.1 Generator

The generator follows a 2-dimensional U-Net [4] topology with 5 encoding and 5 decoding layers. Specifically, the architecture of the encoder is  $\text{conv}_{64}\text{-conv}_{128}\text{-conv}_{256}\text{-conv}_{512}\text{-conv}_{1024}$ , where  $\text{conv}_N$  denotes a convolutional layer with  $N$  kernels of size  $3 \times 3$ , stride of 1, and padding of 1. A bias is added to the output, followed by a BatchNorm-2D, and ReLU operation. Between convolutions, there is a 2D MaxPool operation with kernel size and stride both set to 2. The decoder architecture is  $\text{upconv}_{1024}\text{-upconv}_{512}\text{-upconv}_{256}\text{-upconv}_{128}\text{-upconv}_{64}$  where  $\text{upconv}_N$  denotes a convolutional layer with  $N$  kernels which is also upsampled by a scale factor of 2. The kernels for these layers are also  $3 \times 3$  in size and have a stride and padding of both 1. The convolution is also followed by a BatchNorm-2D and ReLU operation.

### 2.2 Discriminator

The discriminator used for adversarial training has the following architecture:  $\text{Disc}_{256}\text{-Disc}_{128}\text{-Disc}_{64}\text{-Disc}_{32}\text{-Disc}_{16}\text{-Disc}_8\text{-Disc}_4$  where  $\text{Disc}_N$  denotes a 2D-convolution with  $N$  kernels of size  $4 \times 4$ , stride of 2, and padding of 1, followed by BatchNorm-2D and leaky ReLU with negative slope of 0.2.  $\text{Disc}_{256}$  is not followed by a batch normalization and in  $\text{Disc}_4$  leaky ReLU is replaced by a sigmoid operation.

### 2.3 Training Methodology and Optimization

The loss functions we use are described in Section 4.2 of our paper. Our losses together are described as:

$$L_G = L_{mae} + \alpha L_{perc} + \beta L_{bce}, \quad (1)$$

where,  $\alpha = 1$ , and  $\beta = 0.1$ .

We detail how we use the L2 perceptual loss here. We utilize a VGG16 model pre-trained on ImageNet [1]. The outputs of three ReLU layers are used: layers 2, 9, and 16.  $\phi_i$  is used to

denote the these layers.  $\phi_1 : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{H/2 \times W/2 \times 64}$ ,  $\phi_2 : \mathbb{R}^{H/2 \times W/2 \times 64} \rightarrow \mathbb{R}^{H/4 \times W/4 \times 128}$ , and  $\phi_3 : \mathbb{R}^{H/4 \times W/4 \times 128} \rightarrow \mathbb{R}^{H/8 \times W/8 \times 256}$ . These outputs are used by the L2 perceptual loss to train the network.

Both the generator and discriminator were trained using the Adam optimizer with  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$  and  $\varepsilon = 1e^{-8}$ . The learning rate for the generator is 0.001 and for the discriminator is 0.0001. We train each of the SIFT, FREAK, and SOSNet networks for 400 epochs each. The first 250 epochs are run without the discriminator contributing to the generator-discriminator combination network. The next 150 epochs are run with both the generator and discriminator losses.

## References

- [1] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [2] Philippe Weinzaepfel, Hervé Jégou, and Patrick Pérez. “Reconstructing an image from its local descriptors”. In: *CVPR*. 2011.
- [3] Emmanuel d’Angelo et al. “From bits to images: Inversion of local binary descriptors”. In: *TPAMI* 36.5 (2013), pp. 874–887.
- [4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *MICCAI*. Springer. 2015.
- [5] Alexey Dosovitskiy and Thomas Brox. “Inverting visual representations with convolutional networks”. In: *CVPR*. 2016.
- [6] Francesco Pittaluga, Sanjeev Koppal, and Ayan Chakrabarti. “Learning privacy preserving encodings through adversarial training”. In: *WACV*. 2019.
- [7] Francesco Pittaluga et al. “Revealing scenes by inverting structure from motion reconstructions”. In: *CVPR*. 2019.