Revealing Scenes by Inverting Structure from Motion Reconstructions

Francesco Pittaluga[1]   Sanjeev J. Koppal[1]   Sing Bing Kang[2]   Sudipta N. Sinha[2]

[1] University of Florida   [2] Microsoft Research

(a) SfM point cloud (top view)   (b) Projected 3D points   (c) Synthesized Image   (d) Original Image
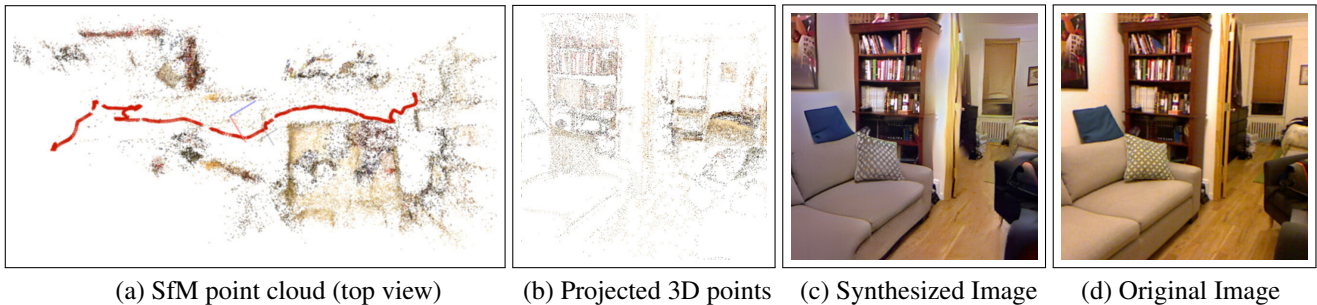
Figure 1: SYNTHESIZING IMAGERY FROM A SFM POINT CLOUD: From left to right: (a) Top view of a SfM reconstruction of an indoor scene, (b) 3D points projected into a viewpoint associated with a source image, (c) the image reconstructed using our technique, and (d) the source image. The reconstructed image is very detailed and closely resembles the source image.

Abstract

Many 3D vision systems localize cameras within a scene using 3D point clouds. Such point clouds are often obtained using structure from motion (SfM), after which the images are discarded to preserve privacy. In this paper, we show, for the first time, that such point clouds retain enough information to reveal scene appearance and compromise privacy. We present a privacy attack that reconstructs color images of the scene from the point cloud. Our method is based on a cascaded U-Net that takes as input, a 2D multi-channel image of the points rendered from a specific viewpoint containing point depth and optionally color and SIFT descriptors and outputs a color image of the scene from that viewpoint. Unlike previous feature inversion methods [46, 9], we deal with highly sparse and irregular 2D point distributions and inputs where many point attributes are missing, namely keypoint orientation and scale, the descriptor image source and the 3D point visibility. We evaluate our attack algorithm on public datasets [24, 39] and analyze the significance of the point cloud attributes. Finally, we show that novel views can also be generated thereby enabling compelling virtual tours of the underlying scene.

1. Introduction

Emerging AR technologies on mobile devices based on ARCore [2], ARKit [3], 3D mapping APIs [1], and new devices such as HoloLens [15] have set the stage for deployment of devices with always-on cameras in our homes, workplaces, and other sensitive environments. Image-based localization techniques allow such devices to estimate their precise pose within the scene [18, 37, 23, 25]. However, these localization methods requires persistent storage of 3D models of the scene which contains sparse 3D point clouds reconstructed using images and SfM algorithms [38].

SfM source images are usually discarded to safeguard privacy. Surprisingly, however, we show that the SfM point cloud and the associated attributes such as color and SIFT descriptors contain enough information to reconstruct detailed comprehensible images of the scene (see Fig. 1 and Fig. 3). This suggests that the persistent point cloud storage poses serious privacy risks that have been widely ignored so far but will become increasingly relevant as localization services are adopted by a larger user community.

While privacy issues for wearable devices have been studied [16], to the best of our knowledge, a systematic analysis of privacy risk of storing 3D point cloud maps has never been reported. We illustrate the privacy concerns by proposing the problem of synthesizing color images from an SfM model of a scene. We assume that the reconstructed model contains a sparse 3D point cloud with optional attributes such as descriptors, color, point visibility and associated camera poses but not the source images.

We make the following contributions: **(1)** We introduce the problem of inverting a sparse SfM point cloud and reconstructing detailed views of the scene from arbitrary viewpoints. This problem differs from the previously studied single-image feature inversion problem due to the need to deal with highly sparse point distributions and a

higher degree of missing information in the input, namely unknown keypoint orientation and scale, unknown image source of descriptors, and unknown 3D point visibilities. **(2)** We present a new approach based on three neural networks where the first network performs visibility estimation, the second network reconstructs the image and the third network uses an adversarial framework to further refine the image quality. **(3)** We systematically analyze variants of the inversion attack that exploits additional attributes that may be available, namely per-point descriptors, color and information about the source camera poses and point visibility and show that even the minimalist representation (descriptors only) are prone to the attack. **(4)** We demonstrate the need for developing privacy preserving 3D representations, since the reconstructed images reveal the scene in great details and confirm the feasibility of the attack in a wide range of scenes. We also show that novel views of the scene can be synthesized without any additional effort and a compelling virtual tour of a scene can be easily generated.

The three networks in our cascade are trained on 700+ indoor and outdoor SfM reconstructions generated from 500k+ multi-view images taken from the NYU2 [39] and MegaDepth [24] datasets. The training data for all three networks including the visibility labels were generated automatically using COLMAP [38]. Next we compare our approach to previous work on inverting image features [46, 9, 8] and discuss how the problem of inverting SfM models poses a unique set of challenges.

2. Related Work

In this section, we review existing work on inverting image features and contrast them to inverting SfM point cloud models. We then broadly discuss image-to-image translation, upsampling and interpolation, and privacy attacks.

Inverting features. The task of reconstructing images from features has been explored to understand what is encoded by the features, as was done for SIFT features by Weinzaepfel et al. [46], HOG features by Vondrick et al. [45] and bag-of-words by Kato and Harada [20]. Recent work on the topic has been primarily focused on inverting and interpreting CNN features [49, 48, 29]. Dosovitskiy and Brox proposed encoder-decoder CNN architectures for inverting many different features (DB1) [9] and later incorporated adversarial training with perceptual loss functions (DB2) [8]. While DB1 [9] showed some qualitative results on inverting sparse SIFT, both papers focused primarily on dense features. In contrast to these feature inversion approaches, we focus solely on inverting SIFT descriptors stored along with SfM point clouds. While the projected 3D points on a chosen viewpoint may resemble single image SIFT features, there are some key differences. First, our input 2D point distributions can be highly sparse and irregular, due to the

typical inherent sparsity of SfM point clouds. Second, the SIFT keypoint scale and orientation are unknown since SfM methods retain only the descriptors for the 3D points. Third, each 3D point typically has only one descriptor sampled from an arbitrary source image whose identity is not stored either, entailing descriptors with unknown perspective distortions and photometric inconsistencies. Finally, the 3D point visibilities are also unknown and we will demonstrate the importance of visibility reasoning in the paper.

Image-to-Image Translation. Various methods such as Pix2Pix [19], CycleGan [50], CoGAN [27] and related unsupervised approaches [7, 26, 34] use conditional adversarial networks to transform between 2D representations, such as edge to color, label to color, and day to night images. While such networks are typically dense (without holes) and usually low-dimensional (single channel or RGB), Contour2Im [5] takes sparse 2D points sampled along gradients along with low-dimensional input features. In contrast to our work, these approaches are trained on specific object categories and semantically similar images. While we use similar building blocks to these methods (encoder-decoder networks, U-nets, adversarial loss, and perceptual loss), our networks can generalize to arbitrary images, and are trained on large scale indoor and outdoor SfM datasets.

Upsampling. When the input and output domains are identical, deep networks have shown excellent results on upsampling and superresolution tasks for images, disparity, depth maps and active range maps [4, 28, 43, 36, 17]. However, prior upsampling methods typically focus on inputs with uniform sparsity. Our approach differs due to the non-uniform spatial sampling in the input data which also happens to be high dimensional and noisy since the input descriptors are from different source images and viewpoints.

Novel view synthesis and image-based rendering. Deep networks can significantly improve photorealism in free viewpoint image-based rendering [12, 14]. Additionally, several works have also explored monocular depth estimation and novel view synthesis using U-Nets [11, 24, 31]. Our approach arguably provides similar photorealistic visually quality – remarkably, from sparse SfM reconstructions instead of images. This is disappointing news from a privacy perspective but could be useful in other settings for generating photorealistic images from 3D reconstructions.

CNN-based privacy attacks and defense techniques. Recently, McPherson et al. [30] and Vasiljevic et al. [44] showed that deep models could defeat existing image obfuscation methods. Further more, many image transformations can be considered as adding noise and undoing them as denoising, and here deep networks have been quite successful [47]. To defend against CNN-based attacks, attempts at learning CNN-resistant transformations have shown some

promise [33, 10, 35, 13]. Concurrent to our work, Speciale et al. [41] introduced the privacy preserving image-based localization problem to address the privacy issues we have brought up. They proposed a new camera pose estimation technique using an obfuscated representation of the map geometry which can defend against our inversion attack.

3. Method

The input to our pipeline is a feature map generated from a SfM 3D point cloud model given a specific viewpoint *i.e.* a set of camera extrinsic parameters. We obtain this feature map by projecting the 3D points on the image plane and associating the 3D point attributes (SIFT descriptor, color, etc.) with the discrete 2D pixel where the 3D point projects in the image. When multiple points project to the same pixel, we retain the attributes for the point closest to the camera and store its depth. We train a cascade of three encoder-decoder neural networks for visibility estimation, coarse image reconstruction and the final refinement step which recovers fine details in the reconstructed image.

Visibility Estimation. Since SfM 3D point clouds are often quite sparse and the underlying geometry and topology of the surfaces in the scene are unknown, it is not possible to easily determine which 3D points should be considered as visible from a specific camera viewpoint just using z-buffering. This is because a sufficient number of 3D points may not have been reconstructed on the foreground occluding surfaces. This produces 2D pixels in the input feature maps which are associated with 3D points in the background *i.e.* lie on surfaces which are occluded from that viewpoint. Identifying and removing such points from the feature maps is critical to generating high-quality images and avoiding visual artifacts. We propose to recover point visibility using a data-driven neural network-based approach, which we refer to as VISIBNET. We also evaluate two geometric methods which we refer to as VISIBSPARSE and VISIBDENSE. Both geometric methods however require additional information which might be unavailable.

Coarse Image Reconstructon and Refinement. Our technique for image synthesis from feature maps consists of a coarse image reconstruction step followed by a refinement step. COARSENET is conditioned on the input feature map and produces an RGB image of the same width and height as the feature map. REFINENET outputs the final color image which has the same size, given the input feature map along with the image output of COARSENET as its input.

3.1. Visibility Estimation

If we did not perform explicit visibility prediction in our pipeline, some degree of implicit visibility reasoning would still be carried out by the image synthesis network COARSENET. In theory, this network has access to the input

depths and could learn to reason about visibility. However, in practice, we found that this approach to be inaccurate, especially in regions where the input feature maps contain a low ratio of visible to occluded points. Qualitative examples of these failure cases are shown in Figure 5. Therefore we explored explicit visibility estimation approaches based on geometric reasoning as well as learning.

VisibSparse. We explored a simple geometric method that we refer to as VISIBSPARSE. It is based on the "point splatting" paradigm used in computer graphics. By considering only the depth channel in the input, we apply a $\min$ filter with a $k \times k$ kernel on the feature map to obtain a filtered depth map. Here, we used $k = 3$ based on empirical testing. Each entry in the feature map whose depth value is no greater than 5% of the depth value in the filtered depth map is retained as visible. Otherwise, the point is considered occluded and the associated entry in the input is removed.

VisibDense. When the camera poses for the source images computed during SfM and the image measurements are stored along with the 3D point cloud, it is often possible to exploit that data to compute a dense scene reconstruction. Labatut et al. [21] proposed such a method to compute a dense triangulated mesh by running space carving on the tetrahedral cells of the 3D Delaunay triangulation of the sparse SfM points. We used this method, implemented in COLMAP [38] and computed 3D point visibility based on the reconstructed mesh model using traditional z-buffering.

VisibNet. A geometric method such as VISIBDENSE cannot be used when the SfM cameras poses and image measurements are unavailable. We therefore propose a general regression-based approach that directly predicts the visibility from the input feature maps, where the predictive model is trained using supervised learning. Specifically, we train an encoder-decoder neural network which we refer to as VISIBNET to classify each input point as either "visible" or "occluded". Ground truth visibility labels were generated automatically by leveraging VISIBDENSE on all train, test, and validation scenes. Using VISIBNET's predictions to "cull" occluded points from the input feature maps prior to running COARSENET significantly improves the quality of the reconstructed images, especially in regions where the input feature map contains fewer visible points compared to the number of points that are actually occluded.

3.2. Architecture

A sample input feature map as well as our complete network architecture consisting of VISIBNET, COARSENET, and REFINENET is shown in Figure 2. The input to our network is an $H \times W \times n$ dimensional feature map consisting of n-dimensional feature vectors with different combinations of depth, color, and SIFT features at each 2D location. Except for the number of input/output channels in the
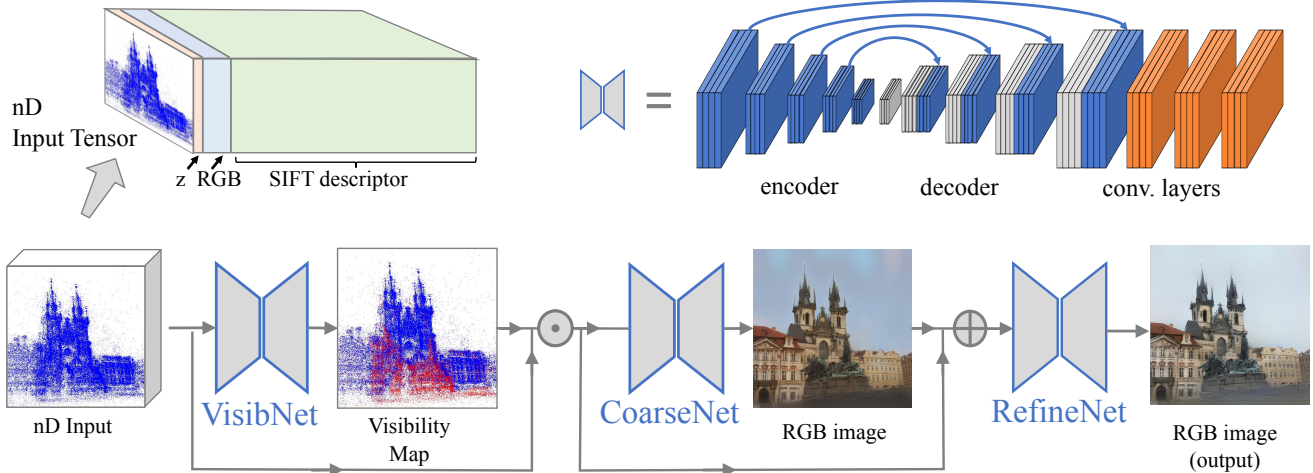
Figure 2: NETWORK ARCHITECTURE: Our network has three sub-networks – VISIBNET, COARSENET and REFINENET. The upper left shows that the input to our network is a multi-dimensional nD array. The paper explores network variants where the inputs are different subsets of depth, color and SIFT descriptors. The three sub-networks have similar architectures. They are U-Nets with encoder and decoder layers with symmetric skip connections. The extra layers at the end of the decoder layers (marked in orange) are there to help with high-dimensional inputs. See the text and supplementary material for details.

first/final layers, each sub-network has the same architecture consisting of U-Nets with a series of encoder-decoder layers with skip connections. Compared to conventional U-Nets, our network has a few extra convolutional layers at the end of the decoder layers. These extra layers facilitate propagation of information from the low-level features, particularly the information extracted from SIFT descriptors, via the skip connections to a larger pixel area in the output, while also helping to attenuate visual artifacts resulting from the highly sparse and irregular distribution of these features. We use nearest neighbor upsampling followed by standard convolutions instead of transposed convolutions as the latter are known to produce artifacts [32].

3.3. Optimization

We separately train the sub-networks in our architecture, VISIBNET, COARSENET, and REFINENET. Batch normalization was used in every layer, except the final one in each network. We applied Xavier initialization and projections were generated on-the-fly to facilitate data augmentation during training and novel view generation after training. VISIBNET was trained first to classify feature map points as either visible or occluded, using ground-truth visibility masks generated automatically by running VISIBDENSE for all train, test, and validation samples. Given training pairs of input feature maps $F_x \in \mathbb{R}^{H \times W \times N}$ and target source images $x \in \mathbb{R}^{H \times W \times 3}$, VISIBNET's objective is

$$\mathcal{L}_V(x) = -\sum_{i=1}^{M} \big[U_x \log\big((V(F_x)+1)/2\big) + (1 - U_x)\log\big((1 - V(F_x))/2\big) \big]_i, \quad (1)$$

where $V : \mathbb{R}^{H \times W \times N} \to \mathbb{R}^{W \times H \times 1}$ denotes a differentiable function representing VISIBNET, with learnable parameters, $U_x \in \mathbb{R}^{H \times W \times 1}$ denotes the ground-truth visibility map for feature map F_x, and the summation is carried out over the set of M non-zero spatial locations in F_x.

COARSENET was trained next, using a combination of an L1 pixel loss and an L2 perceptual loss (as in [22, 8]) over the outputs of layers *relu1_1*, *relu2_2*, and *relu3_3* of VGG16 [40] pre-trained for image classification on the ImageNet [6] dataset. The weights of VISIBNET remained fixed while COARSENET was being trained using the loss

$$\mathcal{L}_C = ||C(F_x) - x||_1 + \alpha \sum_{i=1}^{3} ||\phi_i(C(F_x)) - \phi_i(x)||_2^2, \quad (2)$$

where $C : \mathbb{R}^{H \times W \times N} \to \mathbb{R}^{H \times W \times 3}$ denotes a differentiable function representing COARSENET, with learnable parameters, and $\phi_1 : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times 64}$, $\phi_2 : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times 128}$, and $\phi_3 : \mathbb{R}^{H \times W \times 3} \to \mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times 256}$ denote the layers *relu1_1*, *relu2_2*, and *relu2_2*, respectively, of the pre-trained VGG16 network.

REFINENET was trained last using a combination of an L1 pixel loss, the same L2 perceptual loss as COARSENET, and an adversarial loss. While training REFINENET, the weights of VISIBNET and COARSENET remained fixed. For adversarial training, we used a conditional discriminator whose goal was to distinguish between real source images used to generate the SfM models and images synthesized by REFINENET. The discriminator trained using cross-entropy loss similar to Eq. (1). Additionally, to stabilize adversarial training, $\phi_1(R(F_x))_1$, $\phi_2(R(F_x))_1$, and

Desc. Src.	Inp. Feat.			MAE			SSIM		
	D	O	S	20%	60%	100%	20%	60%	100%
Si	✓	✓	✓	.126	.105	.101	.539	.605	.631
Si	✓	✓	×	.133	.111	.105	.499	.568	.597
Si	✓	×	✓	.129	.107	.102	.507	.574	.599
Si	✓	×	×	.131	.113	.109	.477	.550	.578
M	✓	×	×	.147	.128	.123	.443	.499	.524

Table 1: INVERTING SINGLE IMAGE SIFT FEATURES: The top four rows compare networks designed for different subsets of single image (Si) inputs: descriptor (D), keypoint orientation (O) and scale (S). Test error (MAE) and accuracy (SSIM) obtained when 20%, 60% and all the SIFT features are used. Lower MAE and higher SSIM values are better. The last row is for when the descriptors originate from multiple (M) different and unknown source images.

$\phi_3(R(F_x))_1$ were concatenated before the first, second, and third convolutional layers of the discriminator as done in [42]. REFINENET denoted as $R()$ has the following loss.

$$\mathcal{L}_R = ||R(F_x) - x||_1 + \alpha \sum_{i=1}^{3} ||\phi_i(R(F_x)) - \phi_i(x)||_2^2 \quad (3)$$
$$+ \beta[\log(D(x)) + \log(1 - D(R(F_x)))].$$

Here, the two functions, $R : \mathbb{R}^{H \times W \times N+3} \rightarrow \mathbb{R}^{H \times W \times 3}$ and $D : \mathbb{R}^{H \times W \times N+3} \rightarrow \mathbb{R}$ denote differentiable functions representing REFINENET and the discriminator, respectively, with learnable parameters. We trained REFINENET to minimize $\mathcal{L}_R$ by applying alternating gradient updates to REFINENET and the discriminator. The gradients were computed on mini-batches of training data, with different batches used to update REFINENET and the discriminator.

4. Experimental Results

We now report a systematic evaluation of our method. Some of our results are qualitatively summarized in Fig. 3, demonstrating robustness to various challenges, namely, missing information in the point clouds, effectiveness of our visibility estimation, and the sparse and irregular distribution of input samples over a large variety of scenes.

Dataset. We use the MegaDepth [24] and NYU [39] datasets in our experiments. MegaDepth (MD) is an Internet image dataset with ∼150k images of 196 landmark scenes obtained from Flickr. NYU contains ∼400k images of 464 indoor scenes captured with the Kinect (we only used the RGB images). These datasets cover very different scene content, image resolution, and generate very different distribution of SfM points and camera poses. Generally, NYU scenes produce far fewer SfM points than the MD scenes.

Preprocessing. We processed the 660 scenes in MD and NYU using the SfM implementation in COLMAP [38]. We

Data	Inp. Feat.			Accuracy		
	z	D	C	20%	60%	100%
MD	✓	×	×	.948	.948	.946
	✓	×	✓	.938	.943	.941
	✓	✓	×	.949	.951	.948
	✓	✓	✓	.952	.952	.950
NYU	✓	×	×	.892	.907	.908
	✓	×	✓	.897	.908	.910
	✓	✓	×	.895	.907	.909
	✓	✓	✓	.906	.916	.917

Table 2: EVALUATION OF VISIBNET: We trained four version of VISIBNET, each with a different set of input attributes, namely, z (depth), D (SIFT) and C (color) to evaluate their relative importance. Ground truth labels were obtained with VisibDense. The table reports mean classification accuracy on the test set for the NYU and MD datasets. The results show that VISIBNET achieves accuracy greater than 93.8% and 89.2% on MD and NYU respectively and is not very sensitive to sparsity levels and input attributes.

partitioned the scenes into training, validation, and testing sets with 441, 80, and 139 scenes respectively. All images of one scene were included only in one of the three groups. We report results using both the average mean absolute error (MAE), where color values are scaled to the range [0,1]. and average structured similarity (SSIM). Note that lower MAE and higher SSIM values indicate better results.

Inverting Single Image SIFT Features. Consider the single image scenario, with trivial visibility estimation and identical input to [9]. We performed an ablation study in this scenario, measuring the effect of inverting features with unknown keypoint scale, orientation, and multiple unknown image sources. Four variants of COARSENET were trained, then tested at three sparsity levels. The results are shown in Table 1 and Figure 4. Table 1 reports MAE and SSIM across a combined MD and NYU dataset. The sparsity percentage refers to how many randomly selected features were retained in the input, and our method handles a wide range of sparsity reasonably well. From the examples in Figure 4, we observe that the networks are surprisingly robust at inverting features with unknown orientation and scale; while the accuracy drops a bit as expected, the reconstructed images are still recognizable. Finally, we quantify the effect of unknown and different image sources for the SIFT features. The last row of Table 1 shows that indeed the feature inversion problem becomes harder but the results are still remarkably good. Having demonstrated that our work solves a harder problem than previously tackled, we now report results on inverting SfM points and their features.

4.1. Visibility Estimation

We first independently evaluate the performance of the proposed VISIBNET model and compare it to the geomet-
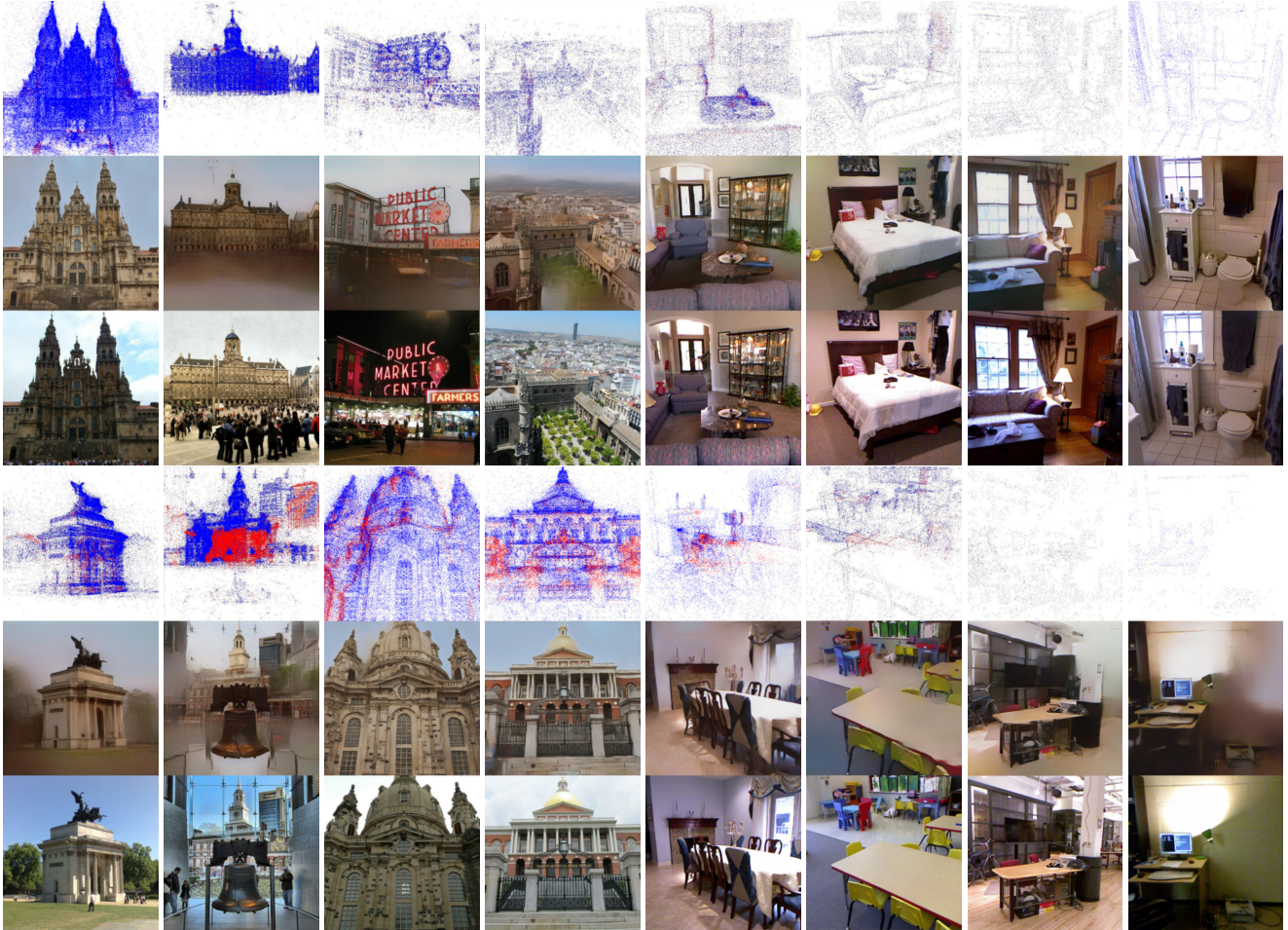
Figure 3: QUALITATIVE RESULTS: Each result is a 3×1 set of square images, showing point clouds (with occluded points in red), image reconstruction and original. The first four columns (top and bottom) show results from the MegaDepth dataset (internet scenes) and the last four columns (top and bottom) show results from indoor NYU scenes. **Sparsity:** Our network handles a large variety in input sparsity (density decreases from left to right). In addition, perspective projection accentuates the spatially-varying density differences, and the MegaDepth outdoor scenes have concentrated points in the input whereas NYU indoor scenes have far samples. Further, the input points are non-homogeneous, with large holes which our method gracefully fills in. **Visual effects:** For the first four columns (MD scenes) our results give the pleasing effect of uniform illumination (see top of first column). Since our method relies on SfM, moving objects are not recovered. **Scene diversity:** The fourth column is an aerial photograph, an unusual category that is still recovered well. For the last four columns (NYU scenes), despite lower sparsity, we can recover textures in common household scenes such as bathrooms, classrooms and bedrooms. The variety shows that our method does not learn object categories and works on any scene. **Visibility:** All scenes benefit from visibility prediction using VISIBNET which for example was crucial for the bell example (lower 2nd column).

ric methods VISIBSPARSE and VISIBDENSE. We trained four variants of VISIBNET designed for different subsets of input attributes to classify points in the input feature map as "visible" or "occluded". We report classification accuracy separately on the MD and NYU test sets even though the network was trained on the combined training set (see Table 2). We observe that VISIBNET is largely insensitive to scene type, sparsity levels, and choice of input attributes such as depth, color, and descriptors. The VISIBNET variant designed for depth only has 94.8% and 89.2% mean

classification accuracy on MD and NYU test sets, respectively, even when only 20% of the input samples were used to simulate sparse inputs. Table 3 shows that when points predicted as occluded by VISIBNET are removed from the input to COARSENET, we observe a consistent improvement when compared to COARSENET carrying both the burdens of visibility and image synthesis (denoted as *Implicit* in the table). While the improvement may not seem numerically large, in Figure 5 we show insets where visual artifacts (bookshelf above, building below) are removed.

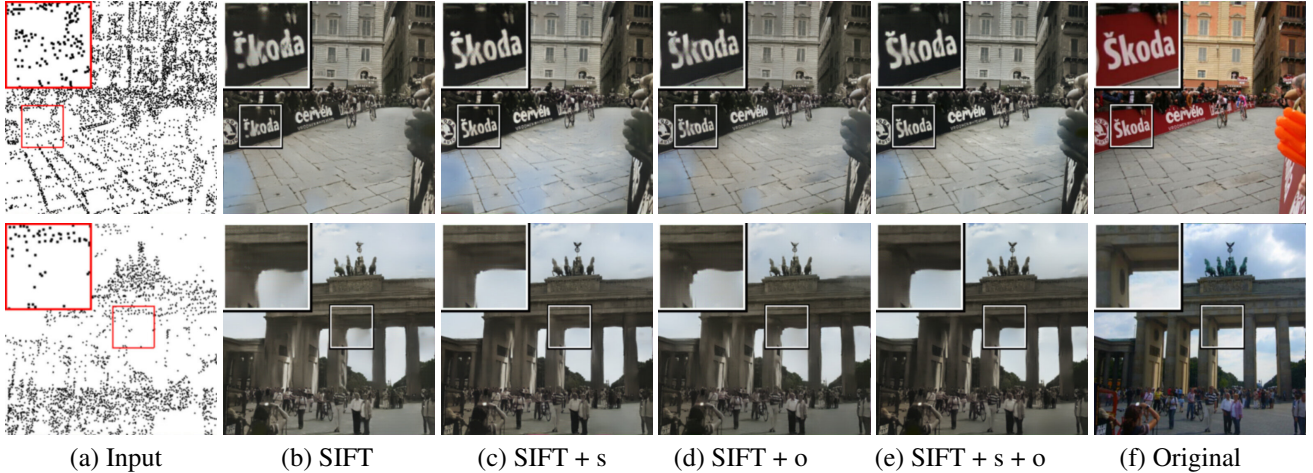| (a) Input | (b) SIFT | (c) SIFT + s | (d) SIFT + o | (e) SIFT + s + o | (f) Original |

Figure 4: INVERTING SIFT FEATURES IN A SINGLE IMAGE: (a) 2D keypoint locations. Results obtained with (b) only descriptor, (c) descriptor and keypoint scale, (d) descriptor and keypoint orientation, (e) descriptor, scale and orientation. (f) Original image. Results from using only descriptors (2nd column) are only slightly worse than the baseline (5th column).

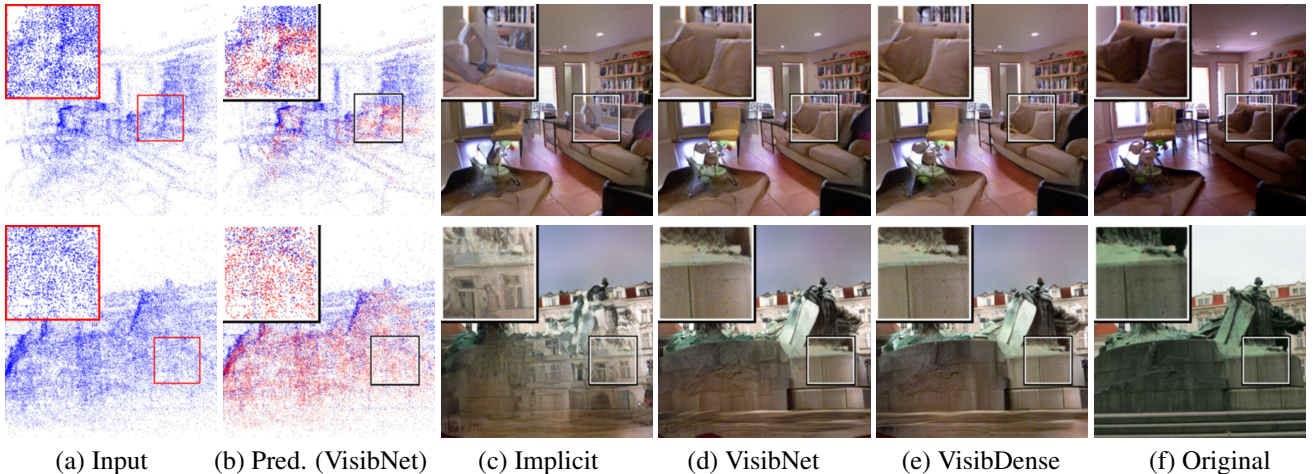| (a) Input | (b) Pred. (VisibNet) | (c) Implicit | (d) VisibNet | (e) VisibDense | (f) Original |

Figure 5: IMPORTANCE OF VISIBILITY ESTIMATION: Examples showing (a) input 2D point projections (in blue), (b) predicted visibility from VISIBNET – occluded (red) and visible (blue) points, (c–e) results from IMPLICIT (no explicit visibility estimation), VISIBNET (uses a CNN) and VISIBDENSE (uses z-buffering and dense models), and (f) the original image.

4.2. Relative Significance of Point Attributes

We trained four variants of COARSENET, each with a different set of the available SfM point attributes. The goal here is to measure the relative importance of each of the attributes. This information could be used to decide which optional attributes should be removed when storing SfM model to enhance privacy. We report reconstruction error on the test set for both indoor (NYU) and outdoor scenes (MD) for various sparsity levels in Table 4 and show qualitative evaluation on the test set in Figure 6. The results indicate that our approach is largely invariant to sparsity and capable of capturing very fine details even when the input feature map contains just depth, although, not surprisingly, color

and SIFT descriptors significantly improves visual quality.

4.3. Significance of RefineNet

In Figure 7 we qualitatively compare two scenes where the feature maps had only depth and descriptors (left) and when it had all the attributes (right). For privacy preservation, these results are sobering. While Table 4 showed that COARSENET struggles when color is dropped (suggesting an easy solution of removing color for privacy), Figure 7 (left) unfortunately shows that REFINENET recovers plausible colors and improves results a lot. Of course, REFINENET trained on all features also does better than COARSENET although less dramatically (Figure 7 (right)).

Data	Visibility Est.	MAE			SSIM		
		20%	60%	100%	20%	60%	100%
MD	Implicit	.201	.197	.195	.412	.436	.445
	VisibSparse	.202	.197	.196	.408	.432	.440
	VisibNet	.201	.196	.195	.415	.440	.448
	VisibDense	.201	.196	.195	.417	.442	.451
NYU	Implicit	.121	.100	.094	.541	.580	.592
	VisibSparse	.122	.100	.094	.539	.579	.592
	VisibNet	.120	.098	.092	.543	.583	.595
	VisibDense	.120	.097	.090	.545	.587	.600

Table 3: IMPORTANCE OF VISIBILITY ESTIMATION: Both sub-tables show results obtained using IMPLICIT *i.e.* no explicit occlusion reasoning where of burden of visibility estimation implicitly falls on COARSENET, *VisibNet* and the geometric methods VISIBSPARSE and VISIBDENSE. Lower MAE and higher SSIM values are better.

Data	Inp. Feat.			MAE			SSIM		
	z	D	C	20%	60%	100%	20%	60%	100%
MD	✓	×	×	.258	.254	.253	.264	.254	.250
	✓	×	✓	.210	.204	.202	.378	.394	.403
	✓	✓	×	.228	.223	.221	.410	.430	.438
	✓	✓	✓	.201	.196	.195	.414	.439	.448
NYU	✓	×	×	.295	.290	.289	.244	.209	.197
	✓	×	✓	.148	.121	.111	.491	.528	.546
	✓	✓	×	.207	.179	.171	.493	.528	.539
	✓	✓	✓	.121	.099	.093	.542	.582	.594

Table 4: EFFECT OF POINT ATTRIBUTES: Performance of four networks designed for different sets of input attributes – z (depth), D (SIFT) and C (color), on MD and NYU. Input sparsity is simulated by applying random dropout to input samples during training and testing.
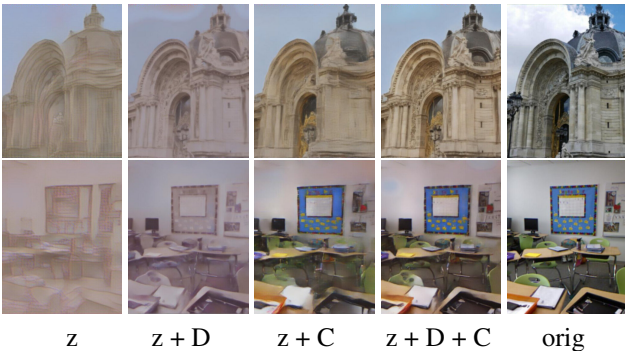

z   z + D   z + C   z + D + C   orig

Figure 6: EFFECT OF POINT ATTRIBUTES: Results obtained with different attributes. Left to right: depth [z], depth + SIFT [z + D], depth + color [z + C], depth + SIFT + color [z + D + C] and the original image. (see Table 4).

4.4. Novel View Synthesis

Our technique can be used to easily generate realistic novel views of the scene. While quantitatively evaluating

z + D   z + D + C

Figure 7: IMPORTANCE OF REFINENET: (Top row) COARSENET results. (Bottom Row) REFINENET results. (Left) Networks use depth and descriptors (z + D). (Right) Networks use depth, descriptor and color (z + D + C).

Figure 8: NOVEL VIEW SYNTHESIS: Synthesized images from virtual viewpoints in two NYU scenes [39] helps to interpret the cluttered scenes (see supplementary video).

such results is more difficult (in contrast to our experiments where aligned real camera images are available), we show qualitative results in Figure 8 and generate virtual tours based on the synthesized novel views[1]. Such novel view based virtual tours can make scene interpretation easier for an attacker even when the images contain some artifacts.

5. Conclusion

In this paper, we introduced a new problem, that of inverting a sparse SfM point cloud and reconstructing color images of the underlying scene. We demonstrated that surprisingly high quality images can be reconstructed from the limited amount of information stored along with sparse 3D point cloud models. Our work highlights the privacy and security risks associated with storing 3D point clouds and the necessity for developing privacy preserving point cloud representations and camera localization techniques, where the persistent scene model data cannot easily be inverted to reveal the appearance of the underlying scene. This was also the primary goal in concurrent work on privacy preserving camera pose estimation [41] which proposes a defense against the type of attacks investigated in our paper. Another interesting avenue of future work would be to explore privacy preserving features for recovering correspondences between images and 3D models.

---

[1] see the video in the supplementary material.

References

[1] 6D.AI. http://6d.ai/, 2018.

[2] ARCore. developers.google.com/ar/, 2018.

[3] ARKit. developer.apple.com/arkit/, 2018.

[4] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich. Estimating depth from RGB and sparse sensing. In *ECCV*, pages 167–182, 2018.

[5] T. Dekel, C. Gan, D. Krishnan, C. Liu, and W. T. Freeman. Smart, sparse contours to represent and edit images. In *CVPR*, 2018.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[7] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017.

[8] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[9] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *CVPR*, pages 4829–4837, 2016.

[10] H. Edwards and A. Storkey. Censoring representations with an adversary. In *ICLR*, 2016.

[11] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[12] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, pages 5515–5524, 2016.

[13] J. Hamm. Minimax filter: learning to preserve privacy from inference attacks. *The Journal of Machine Learning Research*, 18(1):4704–4734, 2017.

[14] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (SIGGRAPH Asia Conference Proceedings)*, 37(6), November 2018.

[15] Hololens. https://www.microsoft.com/en-us/hololens, 2016.

[16] J. Hong. Considering privacy issues in the context of google glass. *Commun. ACM*, 56(11):10–11, 2013.

[17] T.-W. Hui, C. C. Loy, and X. Tang. Depth map super-resolution by deep multi-scale guidance. In *ECCV*, 2016.

[18] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, pages 2599–2606, 2009.

[19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017.

[20] H. Kato and T. Harada. Image reconstruction from bag-of-visual-words. In *CVPR*, pages 955–962, 2014.

[21] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multiview reconstruction of large-scale scenes using interest points, Delaunay triangulation and graph cuts. In *ICCV*, pages 1–8, 2007.

[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 4681–4690, 2017.

[23] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, pages 15–29. Springer, 2012.

[24] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[25] H. Lim, S. N. Sinha, M. F. Cohen, M. Uyttendaele, and H. J. Kim. Real-time monocular image-based 6-dof localization. *The International Journal of Robotics Research*, 34(4-5):476–492, 2015.

[26] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[27] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[28] J. Lu and D. Forsyth. Sparse depth super resolution. In *CVPR*, pages 2245–2253, 2015.

[29] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, pages 5188–5196, 2015.

[30] R. McPherson, R. Shokri, and V. Shmatikov. Defeating image obfuscation with deep learning. *arXiv preprint arXiv:1609.00408*, 2016.

[31] M. Moukari, S. Picard, L. Simoni, and F. Jurie. Deep multi-scale architectures for monocular depth estimation. In *ICIP*, pages 2940–2944, 2018.

[32] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[33] F. Pittaluga, S. Koppal, and A. Chakrabarti. Learning privacy preserving encodings through adversarial

training. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 791–799. IEEE, 2019.

[34] X. Qi, Q. Chen, J. Jia, and V. Koltun. Semi-parametric image synthesis. In *CVPR*, pages 8808–8816, 2018.

[35] N. Raval, A. Machanavajjhala, and L. P. Cox. Protecting visual secrets using adversarial nets. In *CV-COPS 2017, CVPR Workshop*, pages 1329–1332, 2017.

[36] G. Riegler, M. Rüther, and H. Bischof. ATGV-Net: Accurate depth super-resolution. In *ECCV*, pages 268–284, 2016.

[37] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *ICCV*, pages 667–674. IEEE, 2011.

[38] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *CVPR*, pages 4104–4113, 2016.

[39] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[41] P. Speciale, J. L. Schönberger, S. B. Kang, S. N. Sinha, and M. Pollefeys. Privacy preserving image-based localization. *arXiv preprint arXiv:1903.05572*, 2019.

[42] D. Sungatullina, E. Zakharov, D. Ulyanov, and V. Lempitsky. Image manipulation with perceptual discriminators. In *ECCV*, pages 579–595, 2018.

[43] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant CNNs. In *International Conference on 3D Vision (3DV)*, pages 11–20, 2017.

[44] I. Vasiljevic, A. Chakrabarti, and G. Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *arXiv preprint arXiv:1611.05760*, 2016.

[45] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Hoggles: Visualizing object detection features. In *CVPR*, pages 1–8, 2013.

[46] P. Weinzaepfel, H. Jégou, and P. Pérez. Reconstructing an image from its local descriptors. In *CVPR*, pages 337–344, 2011.

[47] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, pages 1790–1798, 2014.

[48] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *ICML Workshop on Deep Learning*, 2015.

[49] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833. Springer, 2014.

[50] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *CVPR*, pages 2223–2232, 2017.

Supplementary Material:
Revealing Scenes by Inverting Structure from Motion Reconstructions

Francesco Pittaluga[1]   Sanjeev J. Koppal[1]   Sing Bing Kang[2]   Sudipta N. Sinha[2]

[1] University of Florida   [2] Microsoft Research

A. Implementation Details

In the supplementary material we describe our network architecture and the training procedure in more details.

A.1. Architecture

Our network architecture consists of three sub-networks – VISIBNET, COARSENET and REFINENET. The input to our network is an $H \times W \times n$ dimensional feature map where at each 2D location in the feature map where there is a valid sample, we have one n-dimensional feature vector. This feature vector is obtained by concatenating different subsets of depth, color, and SIFT features which are associated with each 3D point in the SfM point cloud. Except for the number of input/output channels in the first/final layers, each sub-network has the same architecture, that of a U-Net with an encoder and a decoder and with skip connections between the layers in the encoder and decoder networks at identical depths. In contrast to conventional U-Nets, where the decoder directly generates the output, in our network, the output of the decoder is passed through three convolutional layers in sequence to obtain the final output.

More, specifically, the architecture of the encoder is CE256 - CE256 - CE256 - CE512 - CE512 - CE512, where CEN denotes a convolutional layer with N kernels of size 4×4 and stride equal to 2 followed by an addition of a bias, batch normalization, and a ReLU operation.

The architecture of the decoder is CD512 - CD512 - CD512 - CD256 - CD256 - CD256 - C128 - C64 - C32 - C3, where CDN denotes nearest neighbor upsampling by a factor of 2 followed by a convolutional layer with N kernels of size 3×3 and a stride equal to 1, followed by an addition of a bias, batch normalization, and a ReLU operation. CN layers are the same as CDN layers but without the upsampling operation. In the final layer of the decoder, the ReLU is replaced with a tanh non-linearity. In REFINENET, all ReLU operations are replaced by leaky ReLU operations in all the layers of the decoder network. In VISIBNET, the final layer of the decoder has 1 kernel instead of 3.

Our discriminator used for adversarial training of REFINENET has the following architecture – CA256 - CA256 - CA256 - CA512 - CA512 - CA512 - FC1024 - FC1024 - FC1024 - FC2, where CAN denotes a convolutional layer with N kernels of size 3×3 and stride equal to 1 followed by a 2×2 max pooling operation followed by an addition of a bias, batch normalization, and a leaky ReLU operation. FCN denotes a fully connected layer with N nodes followed by an addition of a bias, and a leaky ReLU operation. In the final layer, the leaky ReLU is replaced by a softmax function.

A.2. Optimization

We used the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$ and a learning rate of 0.0001 for training all networks. Images with resolution 256×256 pixels were used as input to the network during training. However, the trained network was used to process images at a resolution of 512×512 pixels. During training, we resized each image such that the smaller dimension of the resized image was randomly assigned to either 296, 394, or 512 pixels, after which we applied a random 2D cropping to the resized image to obtain a 256×256 image which was the actual input to our network. We used Xavier initialization for all the parameters of our network.

B. Additional Results

We now present qualitative results to show that our network is robust to 2D input which is very sparse. Figure A1 shows two example results. Three images are synthesized on randomly selected 20%, 60% and 100% of all the projected 3D points for the scenes. Despite the high simulated 2D sparsity in the input, the output images are quite interpretable. Figure A2 shows some failure examples.

Supplementary Video. Finally, we encourage the reader to view the supplementary video which makes it easier to visualize the qualitative results shown in the main paper. For two scenes, where the SfM camera poses are available, we show that we can reconstruct the source video by running our method on a frame by frame basis with the camera poses for the source images. Finally, we show results of synthesizing images from novel camera viewpoints. Such results can be used to create virtual tours of the scene, thus making it easier to reveal and the appearance, layout and geometry of the scene.

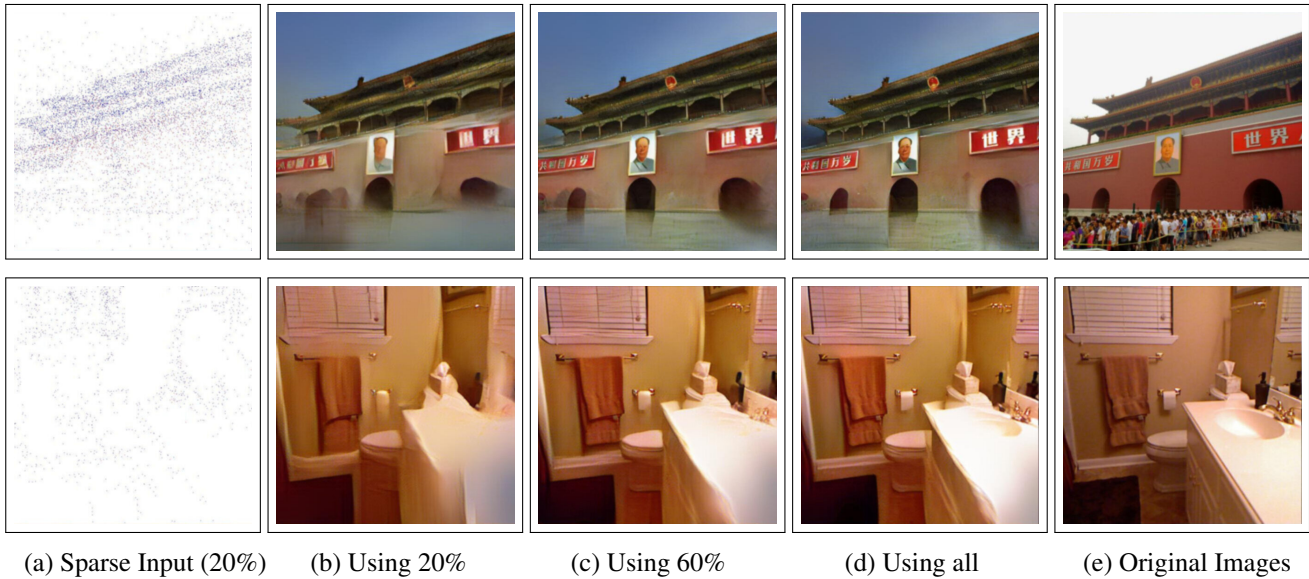| (a) Sparse Input (20%) | (b) Using 20% | (c) Using 60% | (d) Using all | (e) Original Images |

Figure A1: EVALUATING ROBUSTNESS TO SPARSITY: Two sets of images synthesized using our complete pipeline, by running VISIBNET, COARSENET and REFINENET. From left to right: (a) Simulated sparse inputs to our networks. Here, only 20% of the 3D points in the respective SfM models were used. Image synthesized using our method using (b) 20% of the points, (c) 60% of the points, (d) all the points and (e) the original source images. Even when the inputs are extremely sparse, most of the contents of the synthesized images can be easily recognized.

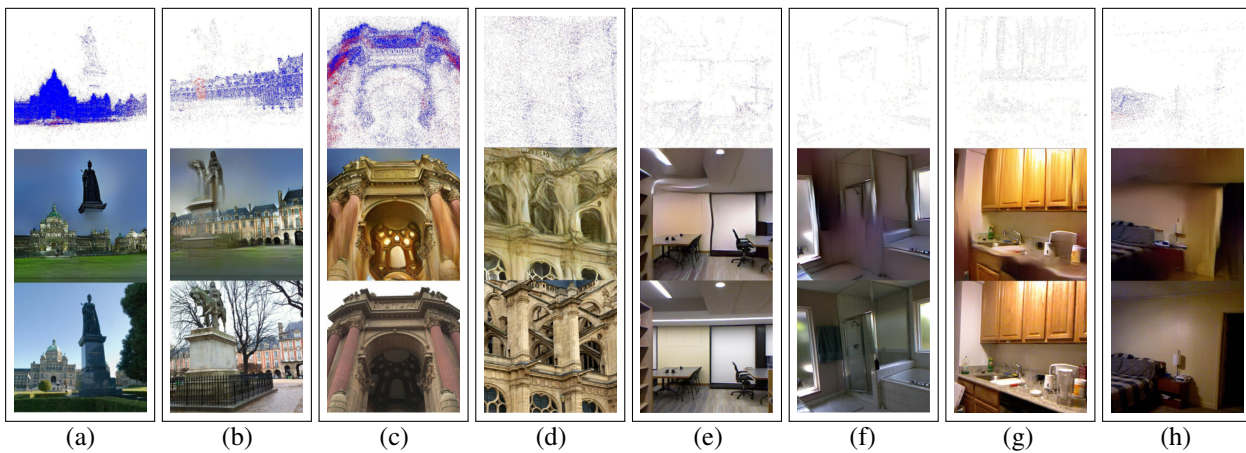| (a) | (b) | (c) | (d) | (e) | (f) | (g) | (h) |

Figure A2: FAILURE EXAMPLES: (a) Dense points on the building in the background overwhelms a few sparse points in the foreground on the base of the statue. VISIBNET in this case incorrectly predicts that the building is visible and this causes the base of the statue to disappear completely in the synthesized image. (b) A similar artifact for a different scene. (c) Parallel straight lines are sometimes poorly handled, such as the lines on the vertical pillars of the monument. (d) The complex occlusions in the architectural structure produce artifacts where the occluded surfaces and the occluders are fused into each other. (e) Straight lines are often reconstructed as curved or bent (f–g) Low sample density in the input common in indoor scenes results in blurry and wavy edges. (h) Finally, spurious 3D points may cause our method to hallucinate structures such as the dark line on the wall which is not actually there.