Learning to Factorize a City Supplemental

Anonymous ECCV submission

Paper ID 2473

"plus ça change, plus c'est la même chose"

(Jean-Baptiste Alphonse Karr)

1 Gamma Correction:

The intrinsic images formula factorize an image **I** into reflectance **R** and shading **S** components.

$$I = R \times S \tag{1}$$

Where I is assumed to be a linear RGB intensity scale. Ideally these linear RGB images should come from high-dynamic range (HDR) exposures because they capture the full spectrum of illumination which are factored into the shading component. However many images, including Google Street View panoramas, are tone-mapped and encoded in the standard RGB (sRGB) colorspace. The tone-mapping procedure involves clipping exposures and gamma correction such that values above the clip are visualized as over-exposed pixels. The gamma correction relates sRGB and linear RGB pixels:

$$I_{\mathsf{sRGB}} = AI^{\gamma} \tag{2}$$

Where $A = 1$ and $\gamma = \frac{1}{2.2}$ typically.

Because tone-mapping irrecoverably loses information about the original lighting, we simply assume the overexposed linear RGB pixels recovered from reversing the gamma correction sRGB are the true brightness. In practice this suffices for intrinsic images as seen by Weiss's MLE Intrinsics [15], Zhou *et al.* [17], and Li and Snavely's BigTime [8] which do gamma "uncorrection" by scaling the sRGB image to take values between [0, 1] and then raising the pixels to the power $\frac{1}{\gamma}$.

However one common trick with manipulating intrinsic images is to work in log-space. More specifically one can take the intrinsic image formula (Eq. 3) and apply the log function to linearize the relationship between log-shading and log-reflectance.

Our observation is that by replacing linear RGB input with the sRGB, the intrinsic image formula becomes:

$$\frac{1}{\gamma}\log(I_{\mathsf{sRGB}}) = \log(R) + \log(S) \tag{3}$$

In log-space, the log-reflectance and log-shading components computed with sRGB and linear RGB are related by a scale $\frac{1}{\gamma}$. In our system, the SPADE decoder G is a learned convolutional neural network that outputs log-shading. Therefore the scale factor $\frac{1}{\gamma}$ can be trivially learned by the convolutional filter weights of the last layer. This has been confirmed as we trained a model using linear RGB and $\frac{1}{\gamma} = 2.2$ and achieved comparable results to ones trained with sRGB. All results shown in submission and main paper have been trained with sRGB inputs **without** extra handling for gamma correction.

One potential problem comes from attempting to decompose images with unconventional gamma curves. We show in Fig. 1 an HDR capture from the Laval Outdoor HDR dataset [2] that has been tone-mapped with various gamma values. While our model performs well near the typical parameter $\gamma = \frac{1}{2.2}$, the

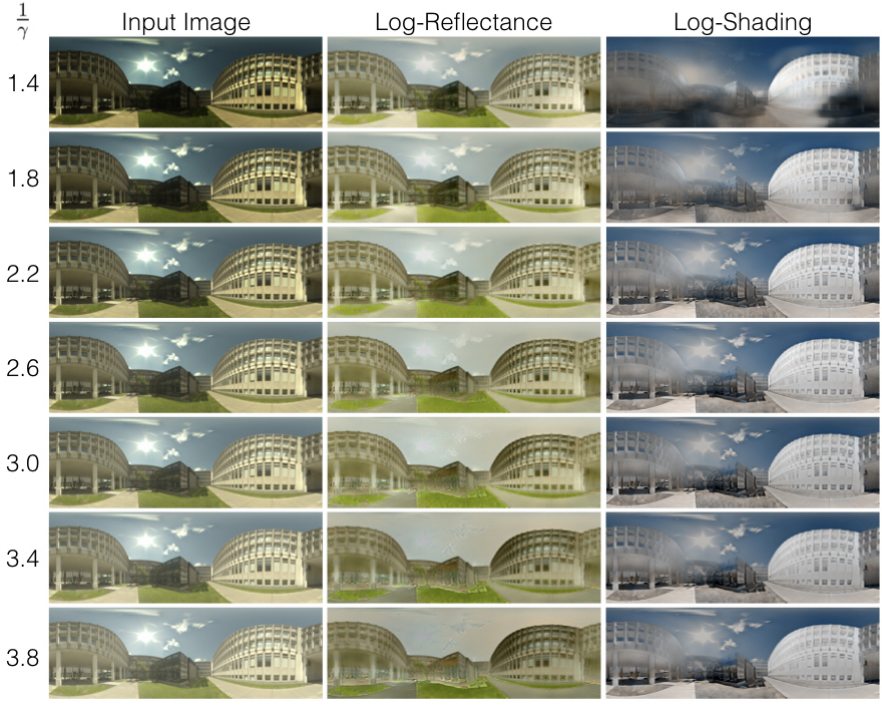

Fig. 1. We show different gamma curves applied to an original HDR panorama capture from the Laval Outdoor HDR dataset [2, 3]. The standard gamma correction is computed with $\frac{1}{\gamma} = 2.2$. For each different value of gamma, we show our model factorizing the resulting gamma corrected image. As shown by the degradation of performance at $\frac{1}{\gamma} = 1.4$, our model is sensitive to decomposing images with unconventional gamma correction.

decomposition degrades especially at $\gamma = \frac{1}{1.4}$. This is an out-of-distribution training problem as most images trained by the model have been corrected with a images that have been tonemapped with the standard $\gamma = \frac{1}{2.2}$. While training with linear RGB images would resolve any ambiguities between different gamma curves, the fundamental problem is that the gamma parameter is unknown and assuming $\gamma = \frac{1}{2.2}$ is a best approximation for most images, in which case the correction can be learned by the decoder.

2 Method

We show in Fig. 2 a diagram of our encoder $\Phi_E, \Phi_\varphi, \Phi_L$. The two illumination descriptor encoders have a shared encoder weights as they are designed to extract transient effects.

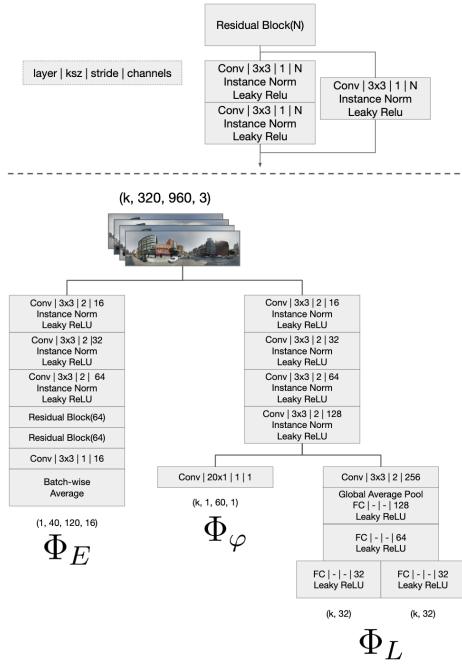

Fig. 2. Our three encoder architectures. The two encoders that output our illumination descriptor, azimuth and lighting context, share the first few convolutional layers before diverging into a horizontal fully-convolutional encoder and a variational encoder.

2.1 Sun Azimuth Encoder

Correction: In the main paper we mistakenly say that φ is a 40-way classification problem. This is a typo and we intended to say that φ is a *60-way classification problem.*

Recall that as a pre-process, all training panoramas are oriented with a consistent heading—the center column faces the same 3D world direction, and the horizon corresponds to the center scanline. In this format, cyclic horizontal translations of the panorama correspond to changing the heading in world coordinates. For example, if a north-facing panorama is shifted by half the length of the image, the resulting panorama will be the same scene, but facing south.

Given the shift-equivariant property of standard convolutions [16], and the equivalence between shifts and rotations for horizon-levelled panoramas, standard convolution operations on panoramas are 3D-yaw-rotation-equivariant. This implies that sun azimuth angle can only be estimated *relative* to the heading of the panoramas, as opposed to predicting an absolute cardinal direction of the sun.

In order to preserve equivariance we must use a fully convolutional encoder network as any global pooling layers would affect the spatial relationship between

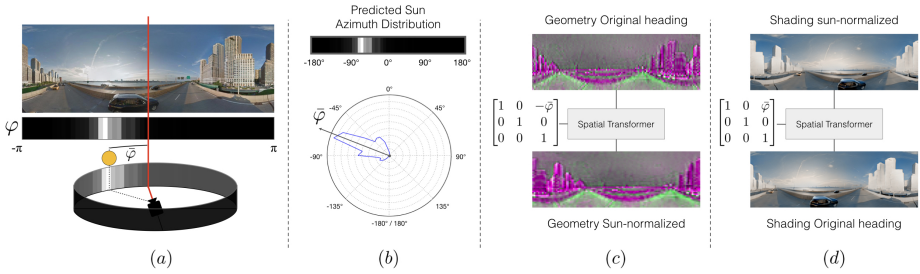

Fig. 3. Our *Panoramic Spatial Transformer* is a novel representational technique for encoding and decoding 1D-rotations (formally SO(2) groups). In (a) we show an input panorama and the output φ of our horizontally fully convolutional encoder Φ_φ. The red line denotes the canonical heading which defines the zero-heading of φ's coordinate system. We assume the canonical heading to always be the center of the panorama. In (b) we visualize the top-down view of the distribution as well as the definition of $\bar{\varphi}$ which is the circular average of the azimuth distribution φ. In (c) we take the geometry representation E and apply a horizontal spatial transformer parameterized by $-\varphi$ which rotates E to the orientation where the sun's position is at the canonical heading. Finally in (d) we show the generated shading with the sun-normalized geometry. To restore the original heading of the image we apply a spatial transformer parameterized by $\bar{\varphi}$ to get the shading of the input panorama.

input and output and result in a network that is invariant (rather than equivariant) to rotations.

The output of the fully convolutional sun azimuth classification network is of dimension 12×60 where 60 is the number of angle classes and the first dimension can be pooled over as it is does not encode angle information to obtain a 1×60 distribution over angles. To this end, we learn a 12×1 learned vertical pooling layer rather than use a fixed max or average pooling operation. The full encoder is thus a horizontally fully-convolutional network, Φ_φ.

2.2 Panoramic Spatial Transformer

We wish to rotate the geometry representation by the predicted sun azimuth such that the sun is always at the same angle (*e.g.* head on). However, the softmax distribution, φ, over the 60 sun azimuth classification buckets spanning $[-\pi, \pi]$ as is described above in Section 2.1 is not differentiable. Luckily, circular angles are ordinal rather than categorical in nature. Therefore, instead of taking a softmax we can compute $\bar{\varphi}$ the circular average, or the expected value of the distribution, in a differentiable way:

$$\bar{\varphi} = \arctan\left(\frac{\mathbf{E}_{\alpha' \sim \varphi}[\sin(\alpha')]}{\mathbf{E}_{\alpha' \sim \varphi}[\cos(\alpha')]}\right) \tag{4}$$

We note that the panoramic rotation operator describes a more general layer that excels at discovering the effects of 1D-rotations. In Fig. 3(a) we show an input

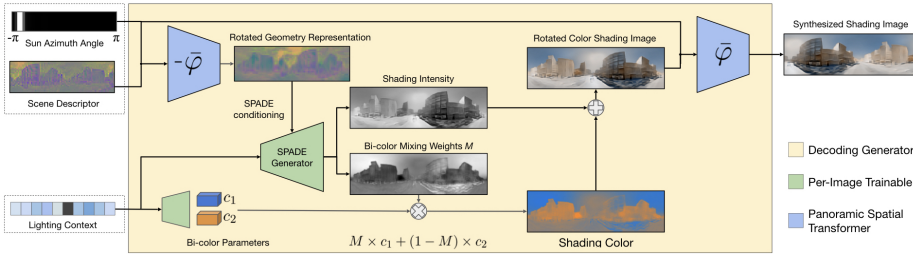

Fig. 4. The decoding generator consists of a Panoramic Spatial Transformer, a SPADE Residual Generator and bi-color shading estimation.

panorama its azimuth distribution produced by Φ_φ. We define the coordinate system of φ relative to a consistent direction across all training examples. This direction is called the *canonical heading* and is indicate by the red line. The canonical heading defines the 0° coordinate with the negative angles to $-\pi$ represent orientations to the left of the canonical heading. Fig. 3(b) shows a polar plot of the distribution of sun azimuth orientations. $\bar{\varphi}$, visualized as the arrow, is about $-70°$ with respect to the canonical heading.

We can use $-\bar{\varphi}$ to parameterize a 1D spatial transformer that rotates the coordinate system to one that is invariant to sun-position as shown in Fig. 3(c). This orientation is called sun-normalization because after rotating by $-\bar{\varphi}$, all geometry representations are oriented with a consistent sun azimuth at 0°. Our insight is that the decoder's job is simplified if the sun is always in the same position. In Fig. 3(d) we show a decoded shading that is sun-normalized. To restore the original heading we use the 1D spatial transformer again parameterized by $\bar{\varphi}$.

By normalizing out sun azimuth, our downstream networks become invariant to sun azimuth, thereby teasing out a disentanglement between the orientation of the sun ($\bar{\varphi}$) and orientation of buildings in the world. Note that this must be used in conjunction with modifications for breaking rotation equivariance described in the next section.

2.3 Generator Architecture:

Our generator is derived from Park *et al.* [12] SPADE residual blocks. More specifically we use noise sampled from the parameters estimated by variational lighting context encoder and condition using SPADE residual blocks described in [12] supplemental. The SPADE residual blocks are conditioned with the azimuth-rotated geometry code described above.

Breaking rotation equivariance. Even though we have rotated our geometry code, the equivariance property described in Sec. 2.1, means that the decoder G will produce the exact same activations as the unrotated version except shifted by the same amount as the original rotation. Therefore we want to break the

rotational equivariance property to allow the network to learn to synthesize different activations for different rotations of the same geometry representations.

To do this, we take one period of a sine and cosine signal from $[-\pi, \pi]$, sample 960 times (or the width of our panoramas), and tile the sampled signal vertically into an image that matches the height of our panoramas. The two images, shown in Fig. 5(SPADE Condition), form a $(320, 960, 2)$ tensor which we resize and concatenate appropriately to the geometry code when its fed into the SPADE conditioning module. While breaking rotational equivariance is important, the implementation of the cosine and sine image can be effectively ignored because it is packaged and self-contained within our SPADE Generator which is **NO LONGER** rotationally equivariant. While we have described an intuitive way to break rotational equivariance, there exists other ways like a learned constant that is concatenated in place of the cosine/sine images.

We show in Fig. 4 the full generator diagram from factors to log-shading output. For information about the intermediate stages that form the bi-color module, please see refer to Sec. 2.4.

Misc. We modify parts of SPADE to account for its new data and usage. We use panorama padding which involves padding with image content from the opposite side of the tensor in the width dimension. To ensure that gradients flow well through the SPADE's condition input, we use average-pooling for downsampling and nearest-neighbors for upsampling. Spectral normalization [11] is used in both the main pathway and the convolutions embedded in the SPADE normalization layers. For explicit details about implementation please see Fig. 5.

2.4 Bi-color shading

Our bi-color shading is very similar to Sunkavalli *et al.* [13] proposed factorized timelapse model. In Sunkavalli *et al.* 's work, they decompose a spatio-temporal timelapse into a sky component, the scene illuminated by diffused sky-radiation, a binary shadow volume, and the scene illuminated by the sun. They alternate minimizing each component to arrive at the final decomposition for a given timelapse. While their results show a very good decomposition, their method suffers from the same issues with intrinsic images in that they cannot transfer appearances across space.

We show in Fig. 4 the different intermediate stages that are used to produce our bi-color shading output. Our SPADE generator G takes in the factors and outputs the *shading intensity* $\in \mathbb{R}^{H,W,1}$ which is equivalent to the classic intrinsic image assumption of gray-scale shading. In addition, G also estimates two global colors $(c_1, c_2) \in \mathbb{R}^3$ and a per-pixel mixing weight $M \in \mathbb{R}^{H,W}$. c_1 and c_2 are learned estimates of sunlight and skylight parameters; typically our model predicts various shades of blue and yellow-orange. These two are mixed using M which is most similar to Sunkavalli *et al.* 's shadow volume. In particular each pixel in M estimates the ratio of sunlight and skylight visible from the scene-point in the world. Our mixed colored representation from combining c_1, c_2, and M defines the *shading color* of the world. Our final outdoor illumination shading is like so:

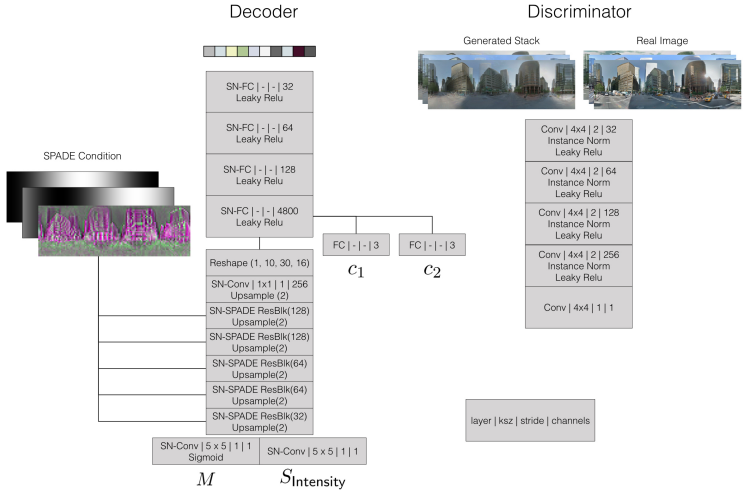

Fig. 5. We show our decoder and discriminator's setup. On left, our decoder uses spectral normalization [11] fully-connected layers to decode global illuminants c_1 and c_2. SPADE Residual Blocks [12] are used to generate the Mask and Shading Intensity. Please refer to [12] for more information about SPADE Residual Blocks. On right we show our PatchGAN discriminator which outputs real and fake logits.

$$\log(S) + c_1 * M + c_2 * (1 - M) \tag{5}$$

In Fig. 6 we show qualitative difference between a mono-color ablation and the full model (bi-color). Note that cast shadow, especially in the third column, are removed from the bi-color reflectance. On the other hand, because the mono-color is incapable of modeling shadow volumes, they leave a strong blue tint behind. The bi-color shading also removes the diffuse sky radiation from the reflectance, leaving a clean plate background that is suitable for relighting as shown in Fig. 7. The residual blue in the mono-color shading model degrades the swap reconstruction when transferring illumination from a different scene.

2.5 Basis Spline Alignment

We propose a novel procedure for dealing with alignment based on image congealing [4]. We find that alignment is an important problem to solve for producing high quality reflectances. We use these reflectances directly when synthesizing relighting scenes, as such alignment directly impacts the quality of our synthesized images.

We'd like to re-iterate that while 3D reconstruction approaches like Martin-Brualla *et al.* [9] and Meshry *et al.* [10] also have to solve for misalignment in input images, their approaches require many hours and hundreds of images to compute camera poses and dense 3D-reconstruction. Further these methods have

Fig. 6. We compare the benefits of the mono-color and bi-color shading assumption. We can see that the mono-color assumption fails to remove shadows completely. There are difficulties with white-balancing as well.

no way to factorize completely unseen scenes. Our proposal enables all of this. We can deal with alignment from a few images and our encoder-decoder allows us to estimate intrinsic factors of unseen scenes from a single image.

Our approach initializes a set of free variables corresponding to control points $\Theta \in [8, 32, 2]$ per image. The control points specify an $8, 32$ grid of horizontal and vertical deformations over an image. These deformations define a forward flow of pixels at the location of the $8, 32$ grid points in the image. To get a full and smoothly changing flow-field, we use a basis-spline (B-spline) to interpolate pixel deformations densely. We can apply the flow-field to warp an image. By warping all images in a stack using their respective Θ, we get an aligned stack.

We use a cubic B-spline surface interpolation to compute the dense flow-field. The basis-spline surface interpolation is a generalization of the 1D B-spline to 2 dimensions corresponding to the height and width of control points. This is not related to the fact that we are learning horizontal and vertical deformations. Each horizontal and vertical deformation are independently computed B-spline surface interpolations.

Because the B-spline is differentiable, we can pass gradients through the B-spline and into the control points. While there exist other family of splines and differentiable warping (thin-plate splines for example), we found that the memory footprint and locally-constrained behavior of control points made B-splines the best candidate for interpolation.

We initialize the control point with noise, however the noise is small such that the initialization is essentially zero. This amounts to initializing the basis spline interpolation with the identity deformation.

2.6 Loss Details

Our primary loss operates on stacks of reflectance and shading outputs produced by our factorization model. Given a timelapse stack $\mathbf{I}$, we factorize the frames to their reflectance stack $\mathbf{R}$ and shading stack $\mathbf{S}$.

Fig. 7. For synthesizing new scenes by swapping lighting context, the bi-color shading model is a necessary improvement. The left column indicates the source weather we wish to copy from. The middle column shows a reconstruction under the mono-color shading. The right column is our bi-color shading. The mono-color fails to correctly synthesize any of the scenes correctly, partly because the reflectance captured is one of a blue sky instead of gray.

Reflectance Consistency Loss. Our reflectance consistency loss enforces the scene albedo to be consistent across time. We do this by minimizing the L_1 inconsistency between every pair of reflectance frames. This loss is used to jointly update alignment parameters and factorization.

$$\mathcal{L}_{\mathsf{RC}} = \sum_{i}^{8} \sum_{j=i+1}^{8} ||\mathbf{R}_i - \mathbf{R}_j||_1 \tag{6}$$

White light penalty. In intrinsic images, there exists a fundamental ambiguity between log-reflectance and log-shading:

$$\log(\mathbf{I}) = (\log(\mathbf{R}) - k) + (\log(\mathbf{S}) + k) \tag{7}$$

where k is an arbitrary channel-shift ambiguity of log-shading that affects the visualization of the components but *does not* affect the resynthesis of the image. Typically this arbitrary shift plays a minor part in the decomposition as the expressiveness of $\log(S)$ is limited to be gray-scale intensity so applying normalization to visualize the components ignores the shift of k in log-reflectance and log-shading. Additionally $\mathcal{L}_{\mathsf{RC}}$, and most approaches in intrinsic images, use shift invariant losses so k cannot influence the optimization.

While we would normally not worry about this, our generator's bi-color shading is expressive enough to produce colored-components c_1, c_2. This means that k ambiguity is an unconstrained color shift. For instance, if we pick an arbitrary color shift (in this case red) to be $k = \epsilon_{\text{red}}$, the generator could learn to predict global color illuminants that have been red-shifted by ϵ_{red}. The result would be a log-shading output that appears red and log-reflectance output that appears cyan. When recombined though the ambiguity cancels and we get a regular image and an identical loss because $\mathcal{L}_{\text{RC}}$ is invariant to k.

For visualization purposes, we impose a white-light loss $\mathcal{L}_{\text{WL}}$ that enforces the average colored illumination c_1, c_2 across time to be white. This effectively encourages the shading generator G to prefer solutions where the color shift ambiguity k is white. As a result we get log-reflectance that appear illuminated under white-light. We take the bi-color augmentation described in Sec. 2.4 $c_1 * M + c_2 * (1 - M)$, denoted as shading color in Fig. 4, to be $\mathbf{B} \in (8, H, W, 3)$ and $\mathbf{B}_i$ to be the bi-color component for frame i.

$$\mathcal{L}_{\text{WL}} = \sum || \sum_{i}^{8} \mathbf{B}_i ||_1 \tag{8}$$

Misc loss. We adopt a standard GAN setup. We use a patch discrminator with 1 scale and 4 layers [5, 14]. Our patch discriminator's only input is the stack reconstruction pixels that have been synthesized from the average reflectance and the predicted shading. Please refer to Fig. 5 for more detail.

We use a hinge adversarial loss $\mathcal{L}_{\text{GAN}}$. $\mathcal{L}_{\text{GAN}}$ appropriately switches between the following generator and discriminator loss when computing gradients of their respective networks. X represent the real images, $D(X)$ represents the discriminator logits, and $F(X)$ represents the encoder-decoder stack reconstruction:

$$\mathcal{L}_{\text{Disc}} = \max(1 + D(X), 0) + \max(1 - D(F(X)), 0) \tag{9}$$

$$\mathcal{L}_{\text{Gen}} = -D(F(X)) \tag{10}$$

We also use a feature-matching loss $\mathcal{L}_{\text{FM}}$ which guides the encoder-decoder to produce images $F(X)$ that are similar to X. This is done by matching the intermediate activations of the discriminator between X and $F(X)$. Let $D_i(*)$ refer to the activations of the i-th layer.

$$\mathcal{L}_{\text{FM}} = \sum_{i} ||D_i(X) - D_i(F(X))||_1 \tag{11}$$

Lastly we include a perceptual loss [6] $\mathcal{L}_{\text{VGG}}$ which also guides the encoder-decoder to produce images $F(X)$ that are "perceptually" similar to the real image X. This is enforced using L_1 loss between real and generated samples' VGG-19 features. We re-use the implementation from [1].

Model	Test-GSV	Laval
Ours	0.806 (9.2°)	0.771 (9.6°)
Supervised Azimuth Encoder	**0.864 (7.92°)**	0.831 (9.3°)
Deep Outdoor Illumination [3]	—	N.A **(4.59°)**

Table 1. Estimating sun azimuth from panoramas. We report the average cosine similarity between prediction and ground truth (higher is better). In parenthesis, the median angular error (lower is better).

2.7 Training Details

We used V100 GPUs with asynchronous gradient updates over a total of $100,000$ stacks each consisting of 8 panoramic images.

Our training stacks are augmented by horizontally translating all their constituent panoramas by the same amount after alignment warping, but before decomposing. This is equivalent to randomly rotating the canonical heading and prevents the model from overfitting to the natural pattern of the sun's position (e.g at higher latitudes in the northern hemisphere, the sun is typically not observed in the geographically north part of the sky).

For both our factorization and discriminator update, we used the default ADAM [7] with learning rate 0.0001 and $\beta_1 = 0$ which was found to work well in SPADE [12]. For learning our warp parameters Θ, we used a lazy implementation of Adam [1] that's optimized for applying efficient sparse updates.

3 Additional Results

3.1 Sun Azimuth Evaluation

We evaluate the goodness of our unsupervised azimuth estimation module by comparing our unsupervised estimates with the true azimuth heading on two test datasets: GSV-TM and Laval Outdoor HDR [2]. The true azimuth for GSV-TM is computed using solar angle equations from the GPS and date-time metadata. Laval panoramas are annotated with azimuth estimated from computing connected components of the brightest pixel.

In order to measure correctness in azimuth estimation, we compute the cosine distance between the predicted and true azimuth angles. Because our azimuth representation is an unsupervised embedding learned by a neural network, the relationship between the output of the encoder and the ground truth sun azimuth angle is ambiguous up to a constant rotation. Therefore, we estimate a rotation of our azimuth representation over a validation set that maximizes the cosine similarity between our finetuned rotated prediction and the real angle of the sun.

We show the full results in Table 1, comparing against a fully supervised azimuth encoder as well as a supervised baseline method [3] on the Laval dataset. We record the average cosine similarity and median angular error (shown in parenthesis) between the prediction and ground truth.

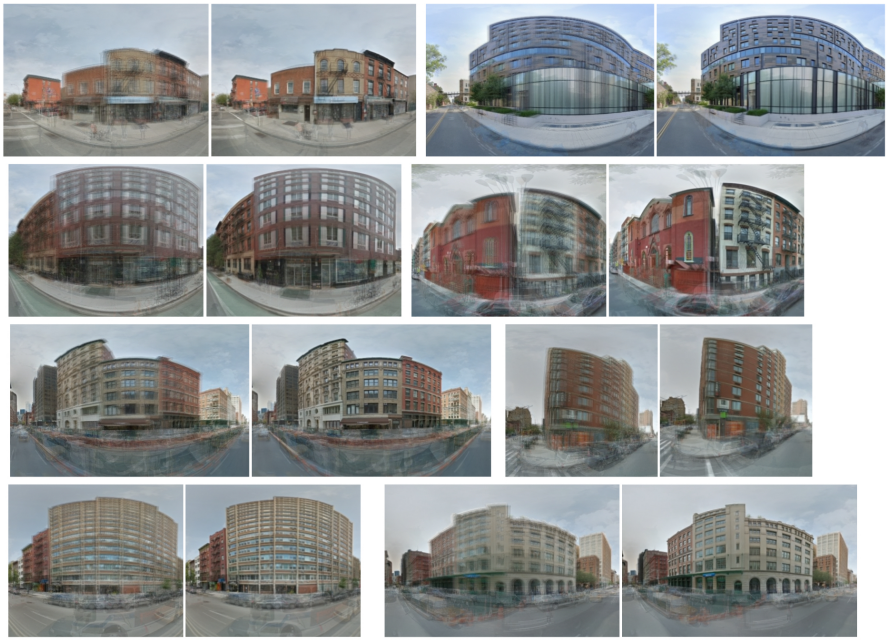

Fig. 8. Additional alignment result. For each pair we show the unaligned average on left and the output of our alignment algorithm on right. These alignments are computed at test-time when the factorization weights are frozen. Given a stack we take ten gradient descent optimization of alignment parameters Θ.

3.2 Alignment Results:

In Fig. 8 we show additional alignment on test stacks. The misaligned stacks highlight the importance of solving alignment for producing high quality images. While we proposed solving for alignment with the factorized reflectance, one could choose to break the feedback loop and solve for alignment using the original RGB pixels. This is equivalent to first solving for alignment as preprocessing to the encoder-decoder.

We show in Fig. 9 that aligning on the original RGB image stack results in poorer alignment than our proposed process. While we did not validate this, we suspect that attempting RGB alignment on even smaller sized stack would result in even more poor results.

3.3 Beyond NYC

Majority of our results were shown on imagery from the test set of NYC GSV-TM. We briefly showed earlier that our factorization works for scenes beyond NYC with intrinsic image decompositions of Paris, London, and Laval Outdoor HDR.

[1] `tf.contrib.opt.LazyAdamOptimizer`

Fig. 9. We show two versions of the basis-spline alignment process on the unaligned stacks in the top row. The middle row shows our resulting alignment average after optimizing for aligning the original RGB pixels. This is equivalent to breaking the alignment-factorize loop by aligning the stacks as a pre-processing step. The bottom row shows our full-model with the alignment-factorize loop. These examples show that aligning with reflectance produces sharper building textures than aligning with RGB. These examples were selected to highlight the discrepancy in alignment results, but even for the median case there are many subtle misalignment problems that aren't immediately obvious.

We include additional decomposition results for each city and also San Francisco in Fig. 11. In addition to the decomposition from the main submission, we also show a stack decomposition comparison with Weiss's MLE Intrinsics [15] and our model on stacks. In the stack decomposition, our full-model has a better signal for removing moving objects like cars and people from both the reflectance and geometry.

4 Applications

4.1 Transferring only Lighting Context

So far we have visualized transferring the whole *illumination descriptor* and manipulating just the azimuth representation φ. To fully demonstrate disentanglement, we also show manipulating just the *lighting context* in Fig. 12(middle). Here we show different scenes with different sun azimuth. For a row, we transfer the same lighting context L while preserving the original sun azimuth. Note how the same bright blue sky is synthesized, but the sun location matches the original scene's sun azimuth.

4.2 Editing Scene Geometry

We show the procedure of inserting objects into different scenes in Fig. 13. This underlying process is how we can synthetically transplant buildings into new

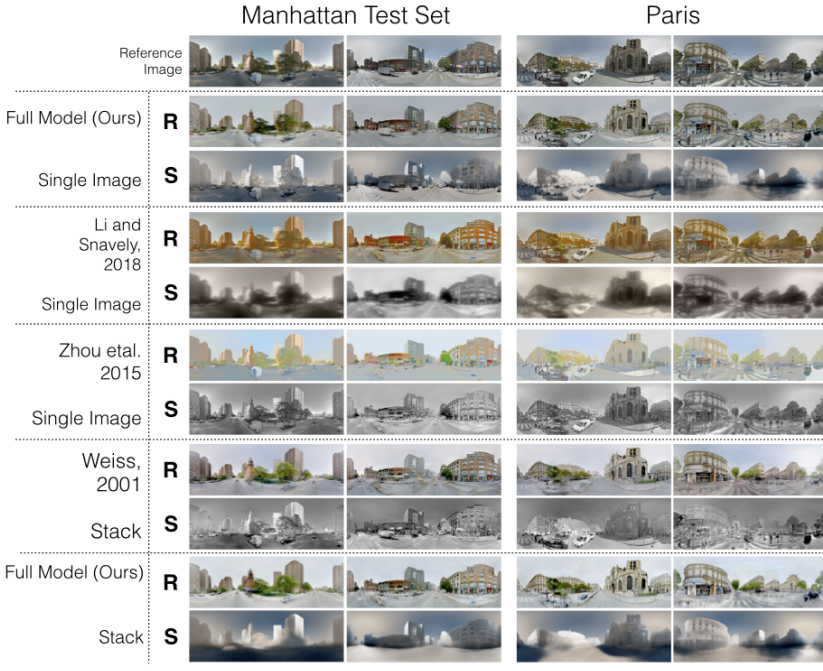

Fig. 10. Additional results from main Fig.6. In addition we also show a stack decomposition comparison between our approach and Weiss's MLE Intrinsics [15]
.

scenes. Additionally the newly synthesized geometry code forms a new scene for which all previously defined factored transformations can be applied. We show scene rotations of the newly spliced building on the attached webpage.

4.3 Hyperlapse Synthesis

We show a hyperlapse synthesis through a section of New York City. In particular the original images often have uncontrollable changes in lighting due to images coming from distinct capture times, resulting in a jarring experience. Using our factorization method, we can normalize for lighting and drive smoothly through Manhattan. Video of the original hyperlapse and adjusted one can be found on the attached webpage.

5 Failure Cases

We show some common failure modes of our factorization in Fig. 15. For single image decompositions, the single biggest source of failures results from poor estimates of our scene descriptors (reflectance and geometry). In the first row, the

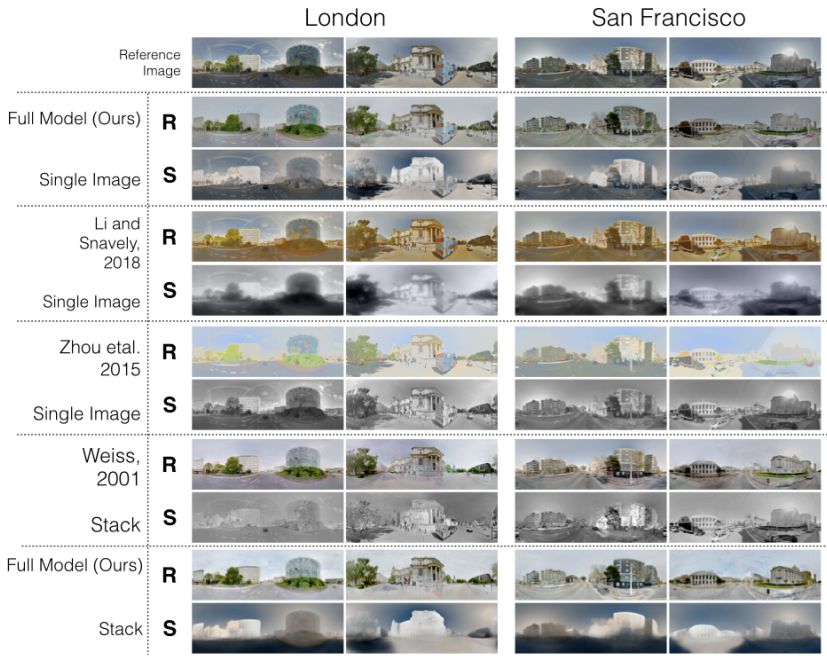

Fig. 11. More decomposition on city scenes from beyond NYC. We show that our factorization generalizes beyond location as shown by our results on London and San Francisco.

single image decomposition struggles to correctly synthesize high-frequency shadows cast by branches. This suggests two things: (1) that multiple views improve the decomposition results by letting the network average out poor geometry and reflectance estimates and (2) the nature of our compressed factorization forces intricate shading interactions like branch shadows to be encoded in the scene descriptor.

The next common failure mode is a result of ghosting of transient objects like cars. While under our current factorization there's no intuitive place to encode cars because they represent changes in the underlying scene geometry that are not permanent. The network learns to average out moving objects in reflectance and attempts to best reconstruct them in the shading images, resulting in wispy gray-scale cars. The second row shows examples of these ghost cars.

Another common failure mode is a poor alignment of timelapse images. Even though images are within a 0.4m radius circle, there are certain scenes where a the basis spline cannot correctly align images due to exceptionally bad parallax. In the last row of Fig. 15 we show timelapses with poor alignment parameters, resulting in an equally bad estimate of reflectance.

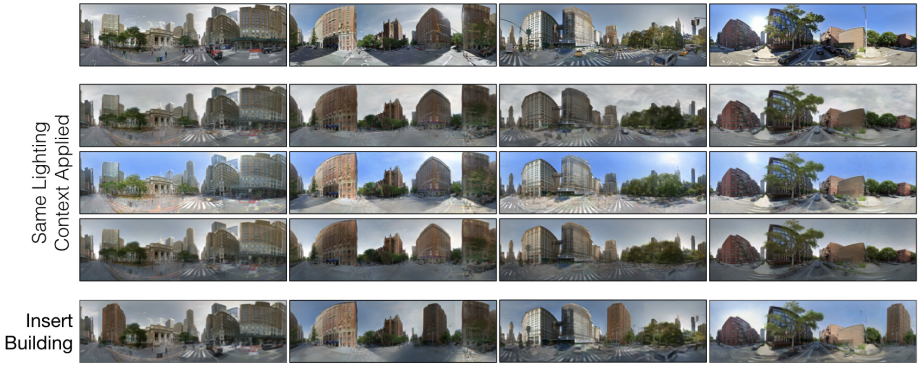

Fig. 12. We show results for applications enabled by our factorization. In the top row we show various scenes we wish to manipulate. The middle section show these scenes relit with a consistent lighting context but original azimuth location. This indicates that we have disentangled sun azimuth from lighting context. In the bottom row we show transplanting a building into the world and updating the lighting realistically.

Based on results from image congealing [4], congealing over larger sets of images results in better chance of aligning images by smoothing the optimization surface. Therefore two possible solutions exists to make alignment better: (1) we can use a smaller baseline to decrease the adverse impact of parallax and (2) we can use more images per stack to smooth the optimization surface when aligning images.

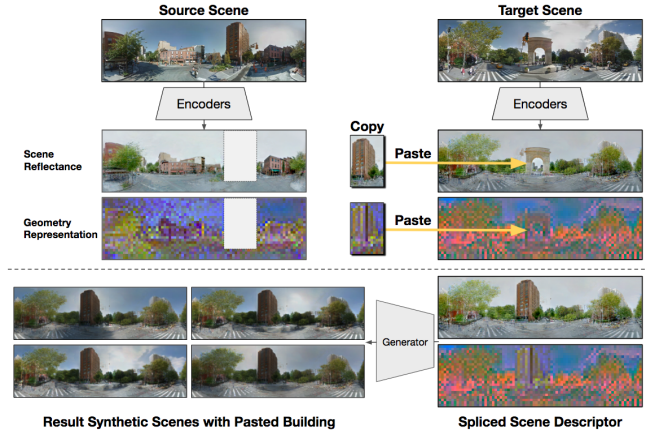

Fig. 13. Our factorized representation lets us copy parts of the scene descriptor and paste them into new scenes before generating realistic-looking modified panoramas. Our synthetically inserted brown building is seamlessly integrated into the target scene.

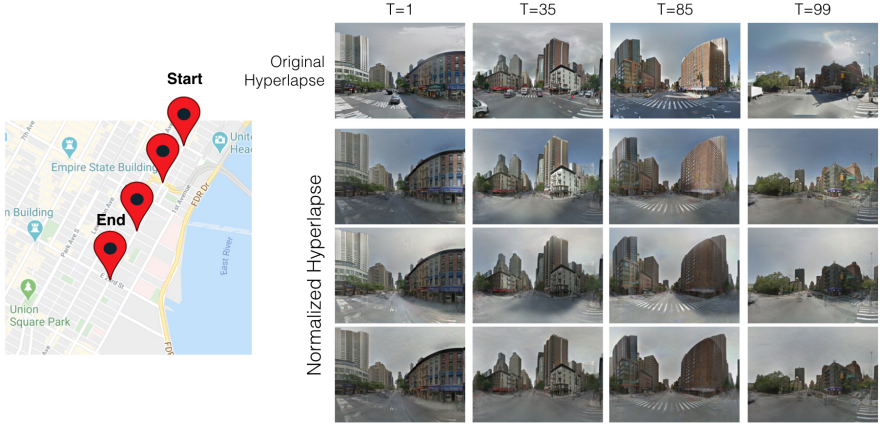

Fig. 14. We show a hyperlapse drive down Second Avenue, Manhattan. In the original hyperlapse, the illumination changes frequently resulting in a jarring experience. We show the process of fixing the illumination to make the hyperlapse weather consistent. The full video can be found attached in the supplemental folder.

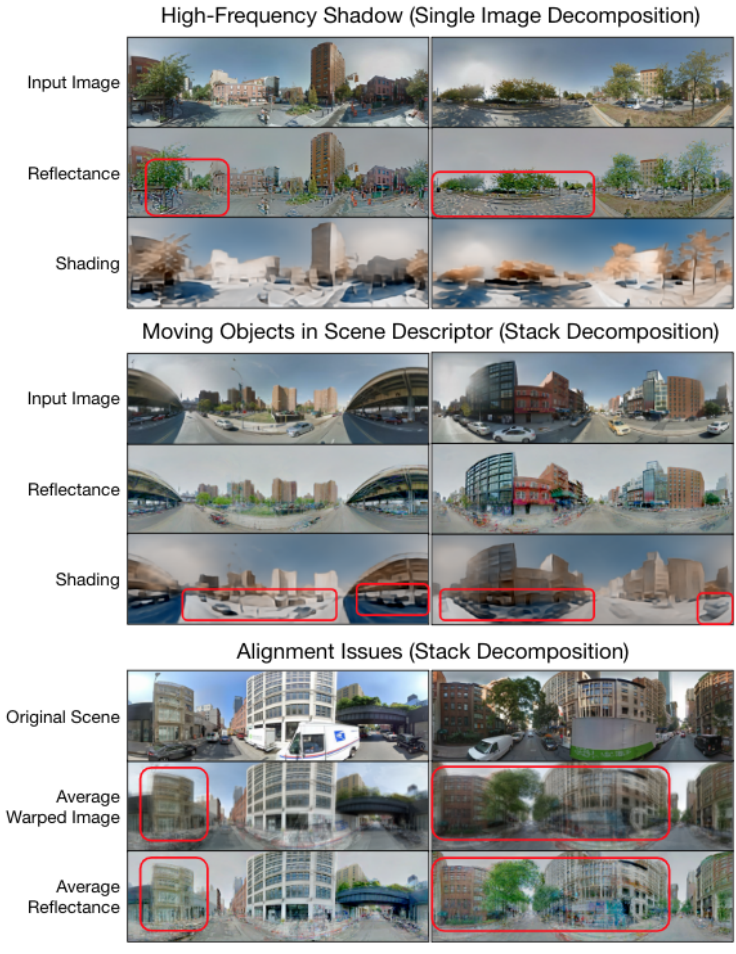

Fig. 15. We show three common failure of our model as well as whether it is associated with a single-image or stack decomposition. In the first row we show how our decomposition fails to correctly remove high-frequency shadows from reflectance like ones left by branches. In the second row we show failures to remove transient objects like cars from the scene descriptor. This results in the synthesis of "ghost" cars. Finally we show a failure of our alignment module. The average images are poorly align, resulting in a poor estimate of the average reflectance.

References

1. Chen, Q., Koltun, V.: Photographic image synthesis with cascaded refinement networks. In: The IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
2. Hold-Geoffroy, Y., Athawale, A., Lalonde, J.: Deep sky modeling for single image outdoor lighting estimation. CoRR **abs/1905.03897** (2019), http://arxiv.org/abs/1905.03897
3. Hold-Geoffroy, Y., Sunkavalli, K., Hadap, S., Gambaretto, E., Lalonde, J.F.: Deep outdoor illumination estimation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
4. Huang, G.B., Jain, V., Learned-Miller, E.: Unsupervised joint alignmemt of complex images. In: International Conference on Computer Vision (ICCV) (2007)
5. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. CVPR 2017 (2016)
6. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision (2016)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014), http://arxiv.org/abs/1412.6980, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015
8. Li, Z., Snavely, N.: Learning intrinsic image decomposition from watching the world. In: Computer Vision and Pattern Recognition (CVPR) (2018)
9. Martin-Brualla, R., Gallup, D., Seitz, S.M.: Time-lapse mining from internet photos. ACM Trans. Graph. **34**(4), 62:1–62:8 (Jul 2015). https://doi.org/10.1145/2766903, http://doi.acm.org/10.1145/2766903
10. Meshry, M., Goldman, D.B., Khamis, S., Hoppe, H., Pandey, R., Snavely, N., Martin-Brualla, R.: Neural rerendering in the wild. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
11. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=B1QRgziT-
12. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
13. Sunkavalli, K., Matusik, W., Pfister, H., Rusinkiewicz, S.: Factored time-lapse video. In: ACM SIGGRAPH 2007 Papers. SIGGRAPH '07, ACM, New York, NY, USA (2007). https://doi.org/10.1145/1275808.1276504, http://doi.acm.org/10.1145/1275808.1276504
14. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional gans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018)
15. Weiss, Y.: Deriving intrinsic images from image sequences. In: International Conference on Computer Vision (ICCV) (2001)
16. Zhang, R.: Making convolutional networks shift-invariant again. CoRR **abs/1904.11486** (2019), http://arxiv.org/abs/1904.11486
17. Zhou, T., Krähenbühl, P., Efros, A.A.: Learning data-driven reflectance priors for intrinsic image decomposition. In: International Conference on Copmuter Vision (ICCV) (2015)