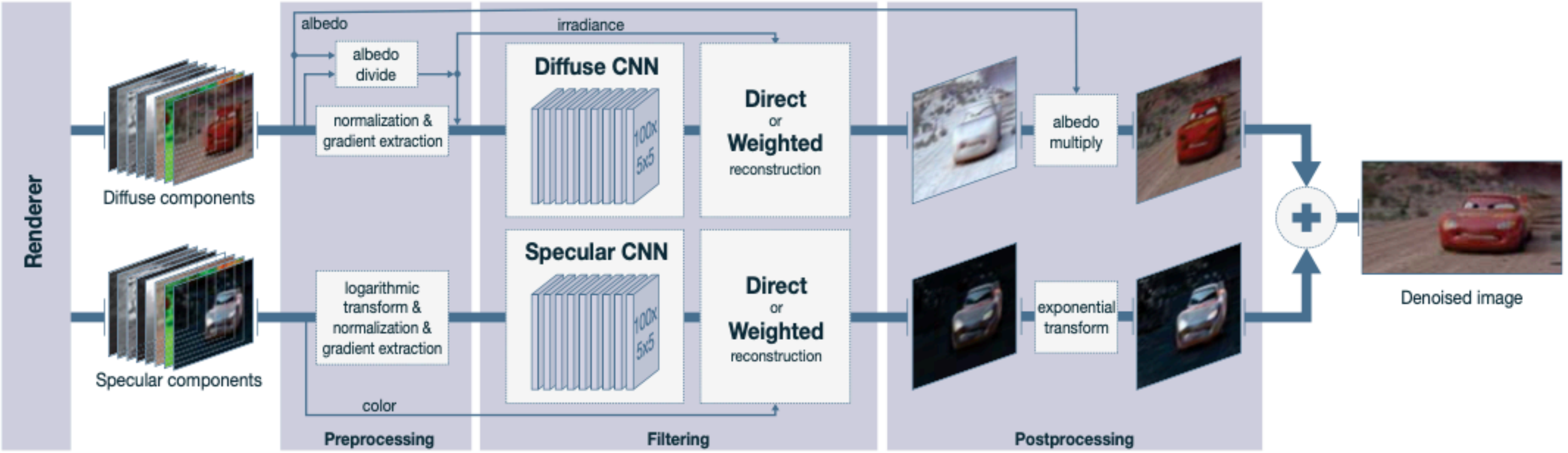
Weakly-Supervised Contrastive Learning in Path Manifold for Monte Carlo Image Reconstruction (WCMC)

SIGGRAPH 2021

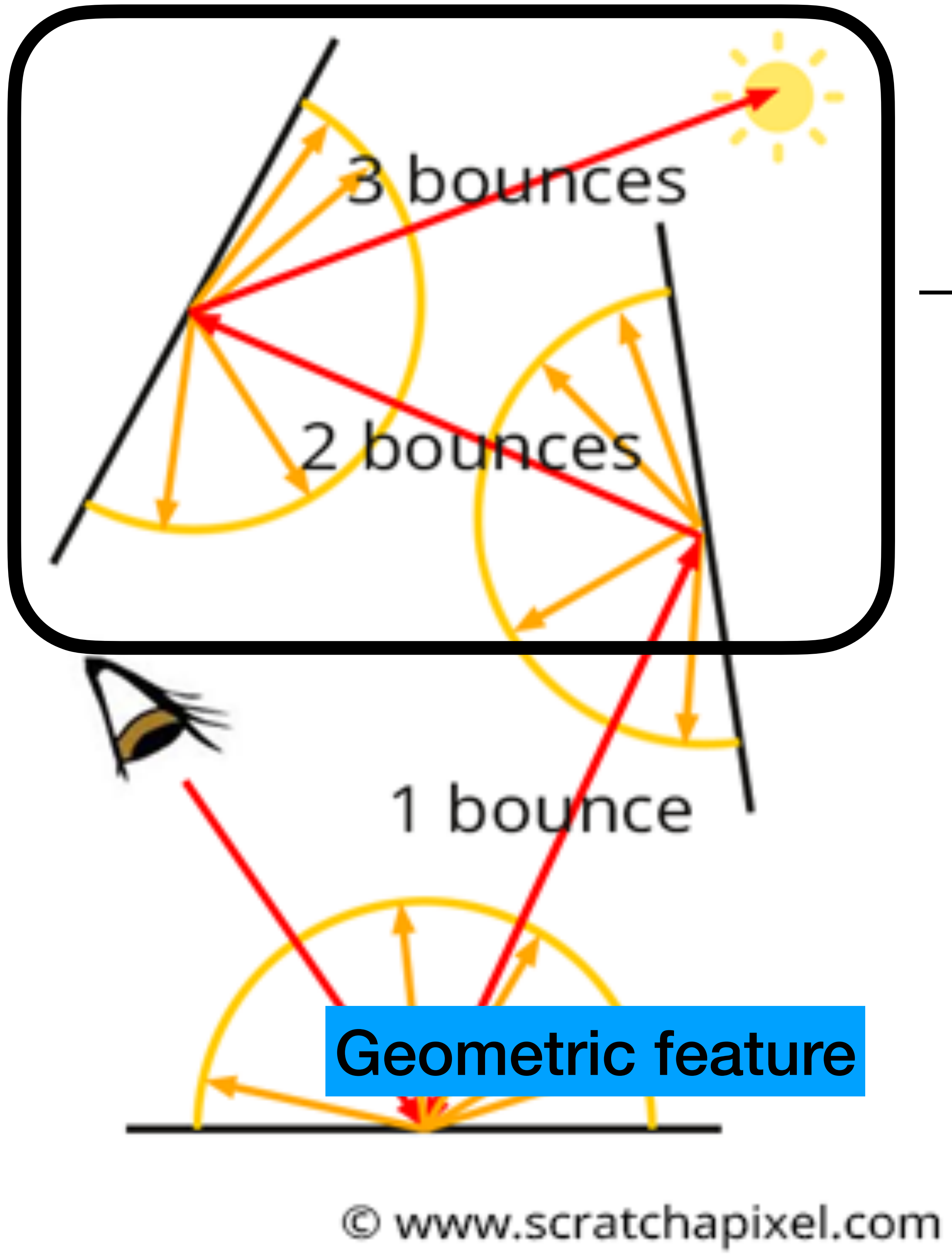
20180475 Seeha Lee

Recap : KPCN



- What was the input?
  - RGB color + geometric features(surface normals, depth, albedo, …)
  - Dimension : 3 + D

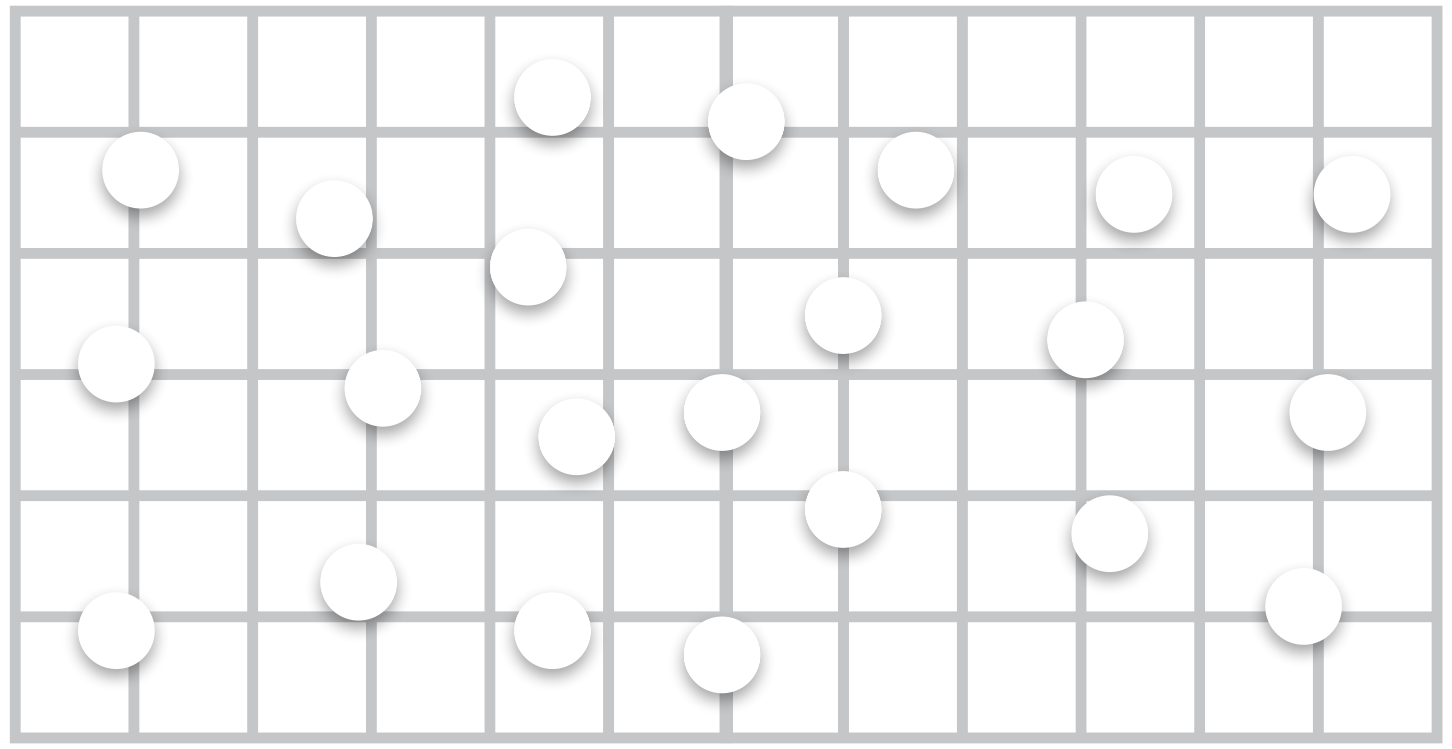
Motivation : Can more information be used?



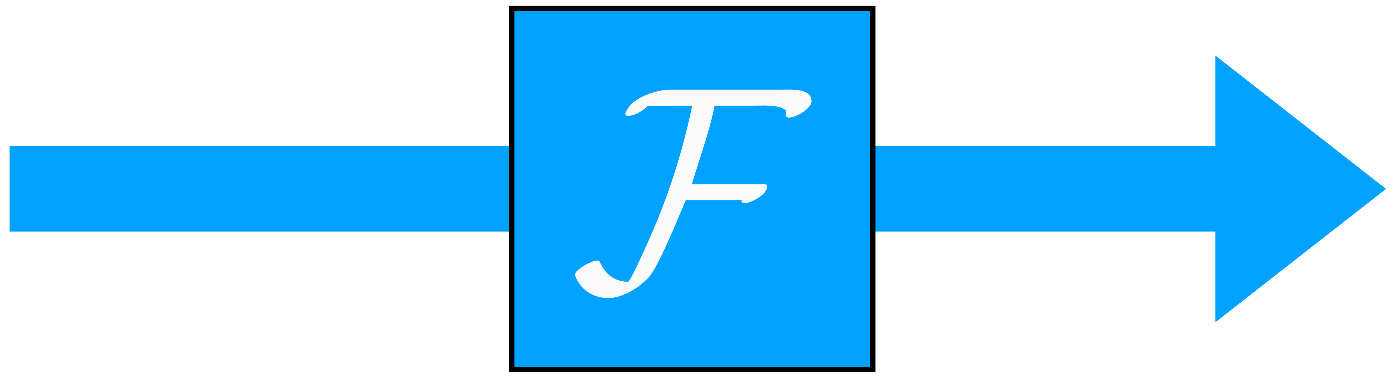
What about the rest?

- Just using geometric feature discards the further information of path tracing due to bounces.
- Why don’t we use this information?

Previous work : SBMC



Input samples
(Radiance + geometric features)



Neural network

Denoised image

- Instead of working in the pixel domain, where the input samples have already been averaged, work with the samples directly
- This paper aims to use even more information

What is the difference?

$$\overline{L}_r(x, \omega_o) = \frac{1}{N} \sum_{i=1}^{N} \frac{L(x, \omega_i) f_s(x, \omega_o, \omega_i) |\cos(\theta_i)|}{q(\omega_i | x, \omega_o)}.$$

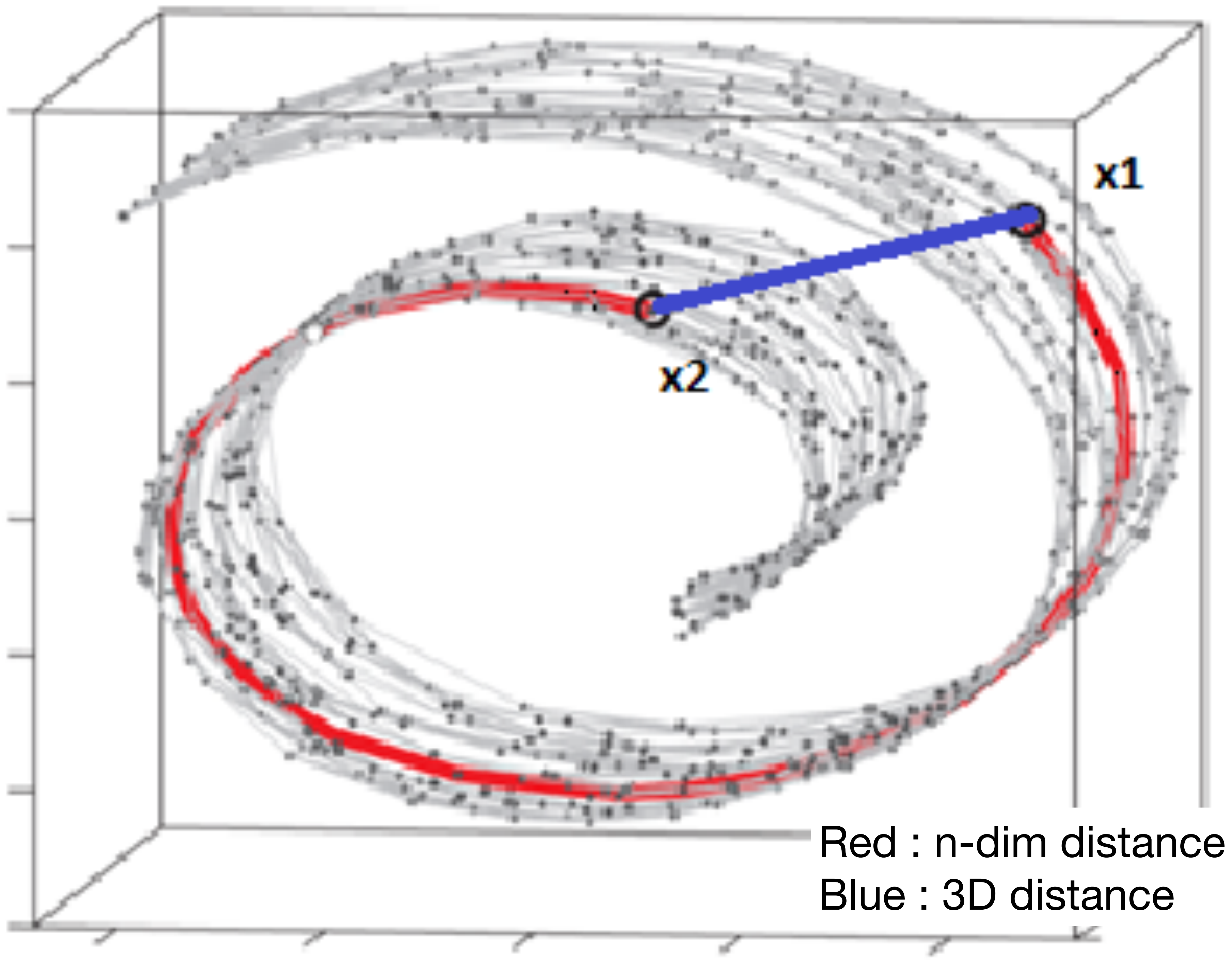
Path based(WCMC) : Use all these information(path features)

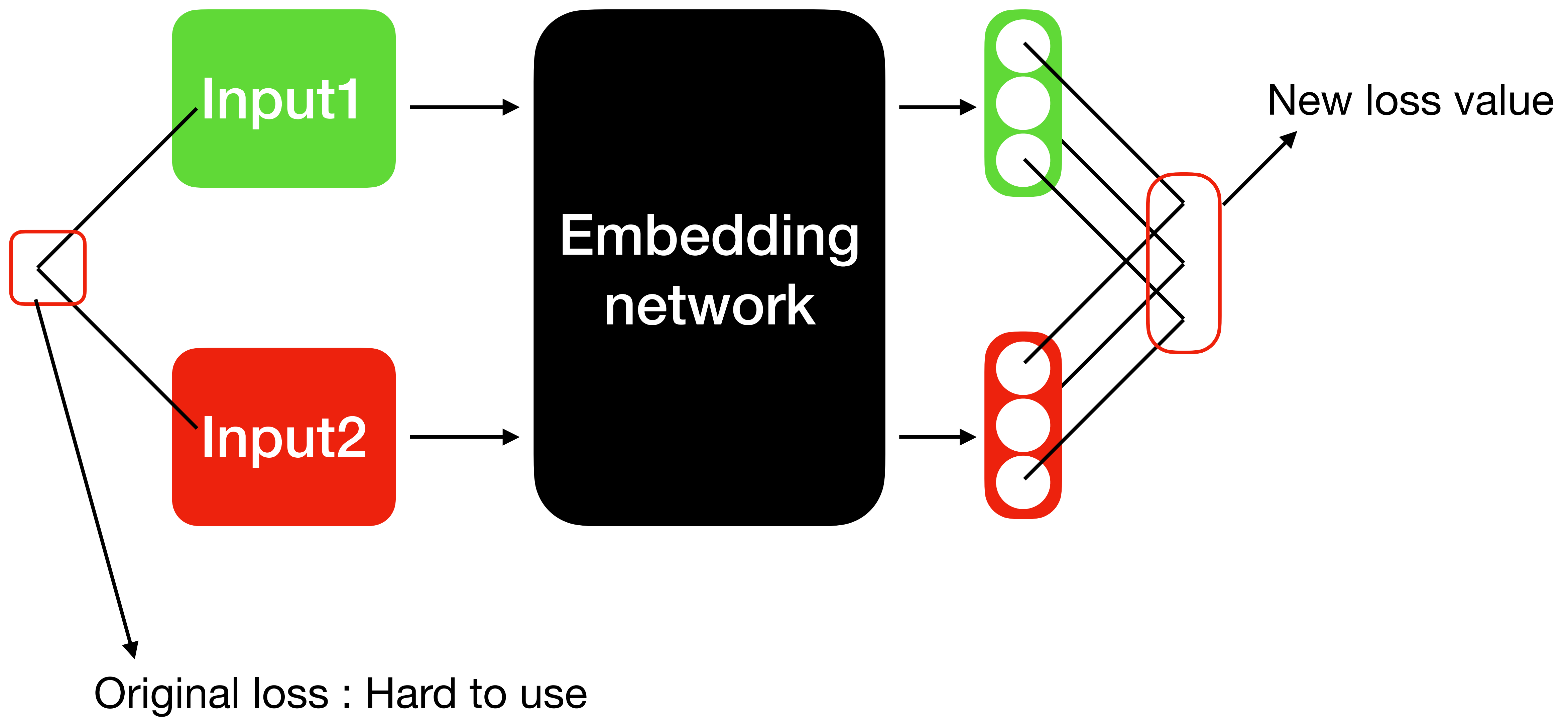
Pixel based(KPCN) : Use only the information of certain pixel

Sample based(SBMC, LBMC) : Use the raw sample information

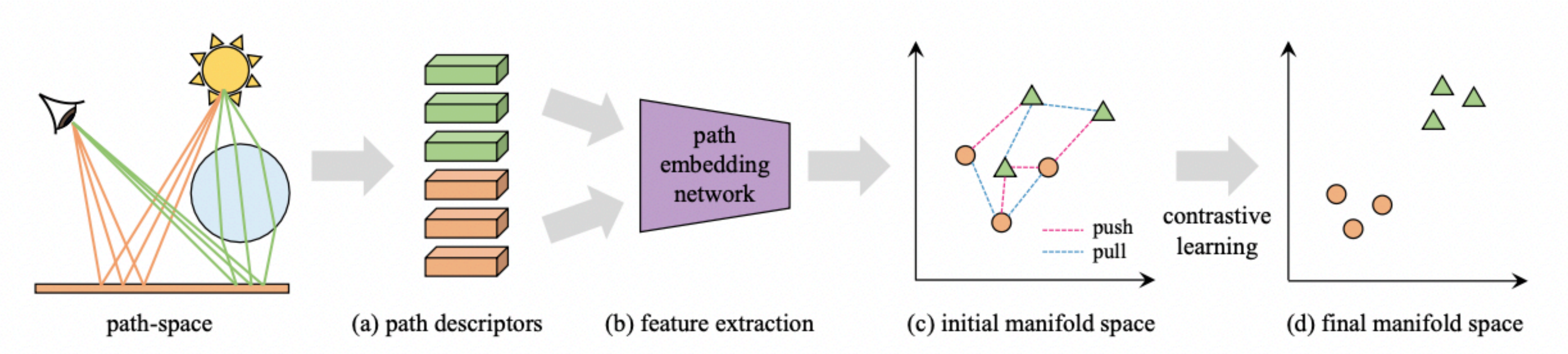
Way to high dimension…

- Path features make the data complex and sparse → Curse of dimension!
- Use manifold!
    - Analyzes the similarity between data to remove redundant dimensions while preserving useful information.





Contrastive learning



- Use contrastive learning to optimize manifold space
- Manipulate distance between embedding pairs according to their label similarities to learn affinity of inputs.
- Use reference radiance as a pseudo label

Path descriptor

Rendering equation :

$$L_r(x,\omega_o) = \int_\Omega L(x,\omega) f_s(x,\omega_o,\omega)\,|\cos(\theta)|\,d\omega, \text{and} \longrightarrow \bar{L}_r(x,\omega_o) = \frac{1}{N}\sum_{i=1}^{N} \frac{L(x,\omega_i) f_s(x,\omega_o,\omega_i)\,|\cos(\theta_i)|}{q(\omega_i|x,\omega_o)}.$$

$$L_o(x,\omega_o) = L_e(x,\omega_o) + L_r(x,\omega_o),$$

$f_s : BSDF \quad L, L_0, L_e, L_r : incidant, outgoing, emitted, reflected\ radiance \quad \omega, \omega_o : incident, outgoing\ direction \quad q : sampling\ density$

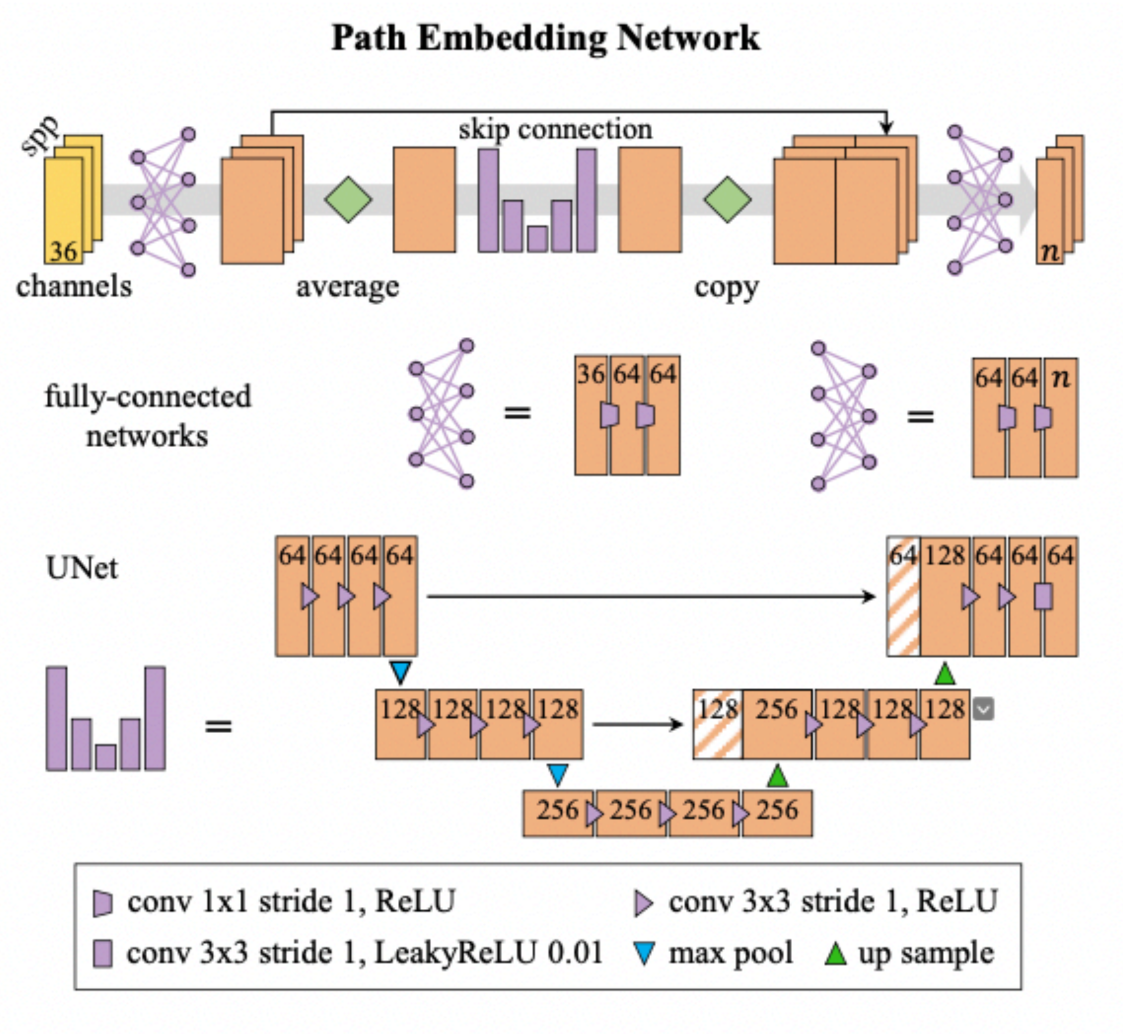
Path descriptor : $\bar{x} = x^{(-1)}x^{(0)}\ldots x^{(k)} \quad x^{(-1)} : eye, x^{(i)} : points$

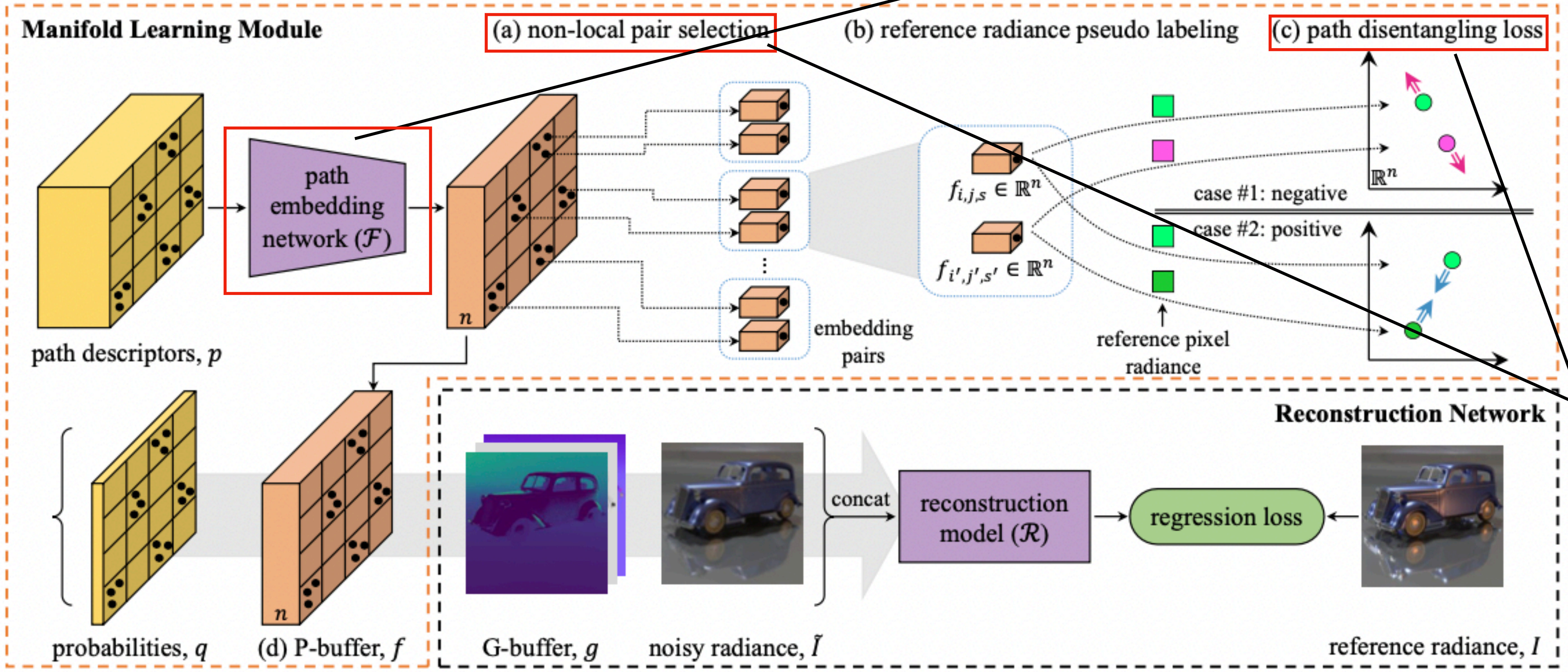
- three channels per vertex for the attenuation; $f_s(x^{(l)},\omega_o^{(l)},\omega^{(l)})\left|\cos(\theta^{(l)})\right|, \forall 0 \le l \le k-1,$
- one channel per vertex for the material-light interaction tag (reflection, transmission, diffuse, glossy, specular),
- one channel per vertex for the roughness parameter of BSDF,
- three channels per path for the radiance undivided by the sampling probability; $L_e(x^{(k)},\omega_o^{(k)}) \prod_{0\le l\le k-1} f_s(x^{(l)},\omega_o^{(l)},\omega^{(l)})\left|\cos(\theta^{(l)})\right|,$
- three channels per path for the photon energy propagated through the path; $L_e(x^{(k)},\omega_o^{(k)}).$
- one channel per path for the path sampling probability, $\prod_{0\le l\le k-1} q(\omega^{(l)}|x^{(l)},\omega_o^{(l)}).$

→ 5k+6 dimension

Overall network

- Uses stacks of fully-connected layers and an UNet to embed path descriptor vectors, considering neighbor paths
- Adopted from Gharbi et al. [2019]





- The network cannot learn meaningful weights if only easy pairs, whose error is relatively smaller than other pairs, are selected. → Use non-local pair selection
- Split the image into patches and create batches. Then select from different patches in the same batch

Algorithm 1 Joint Manifold-Regression Training Algorithm

notations

$\tilde{I}$ and I	noisy input and reference image
g	auxiliary features
p and q	path descriptors and sampling probabilities
$\Theta_{\mathcal{F}}$	weights of the path embedding network
$\Theta_{\mathcal{R}}$	weights of the given reconstruction network
λ	manifold-regression balancing parameter

while total loss is decreasing **do**

$f = \mathcal{F}(p|\Theta_{\mathcal{F}})$ // path embedding

$f', I' = \text{ShuffleWithinBatch}(f, I)$ // non-local pairs

$f'', I'' = \text{ShuffleWithinPatch}(f, I)$ // local pairs

$\hat{I} = \mathcal{R}(\tilde{I}, g, f, q|\Theta_{\mathcal{R}})$ // image reconstruction

$\mathcal{L}_{total} = \lambda(\mathcal{L}_m(f, f', I, I') + \mathcal{L}_m(f, f'', I, I'')) + \mathcal{L}_r(\hat{I}, I)$

$\Theta_{\mathcal{F}}, \Theta_{\mathcal{R}} \leftarrow \text{Adam}(\mathcal{L}_{total})$

end while

return $\Theta_{\mathcal{F}}, \Theta_{\mathcal{R}}$

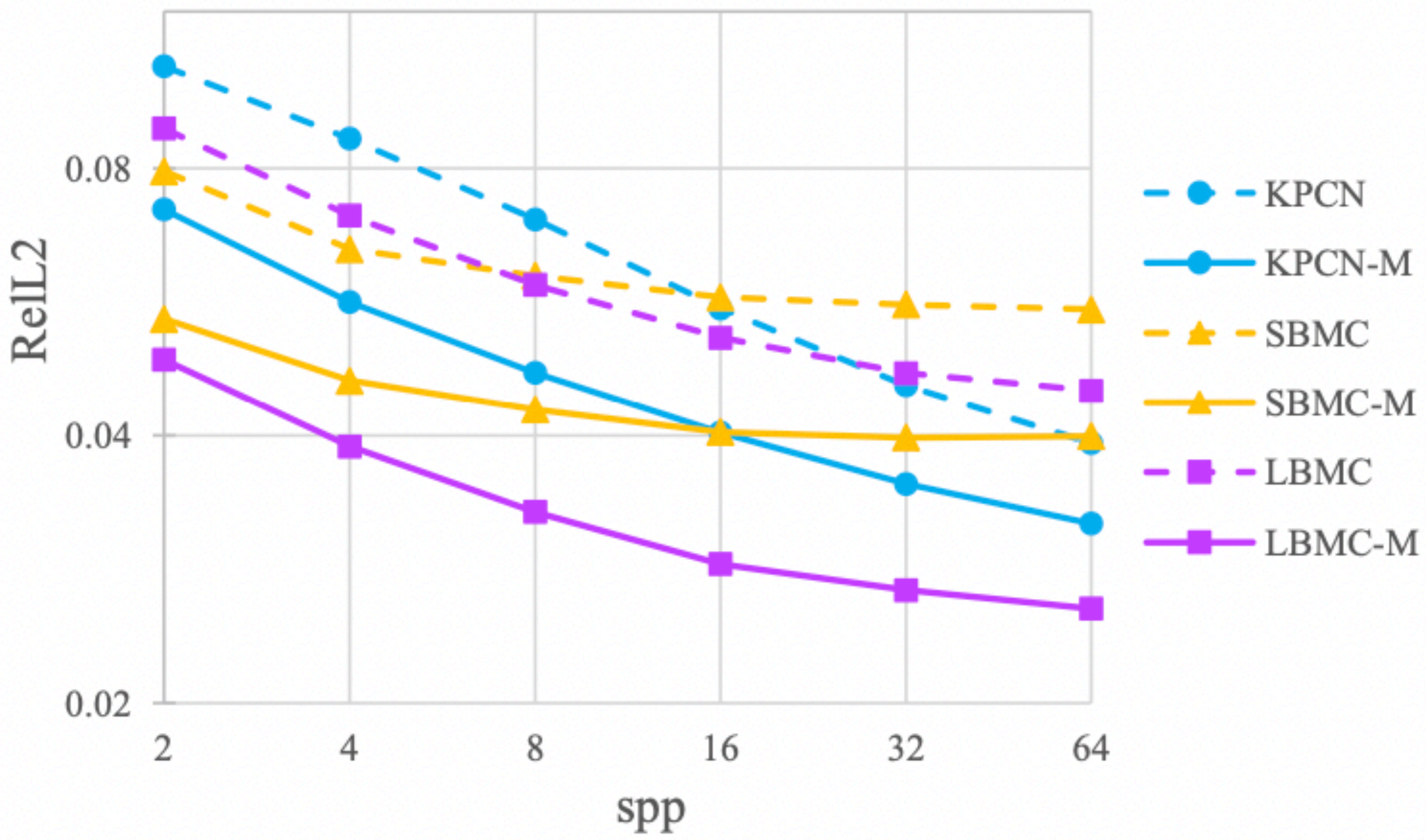
$$\mathcal{L}_m(f_x, f_y, I_x, I_y) = \left(\|f_x - f_y\|_2^2 - \|\tau(I_x) - \tau(I_y)\|_2^2 \right)^2,$$

$\tau(I) = \left(\frac{I}{1+I}\right)^{\gamma}$: Tone mapping function (Reinhard et al. [2002])

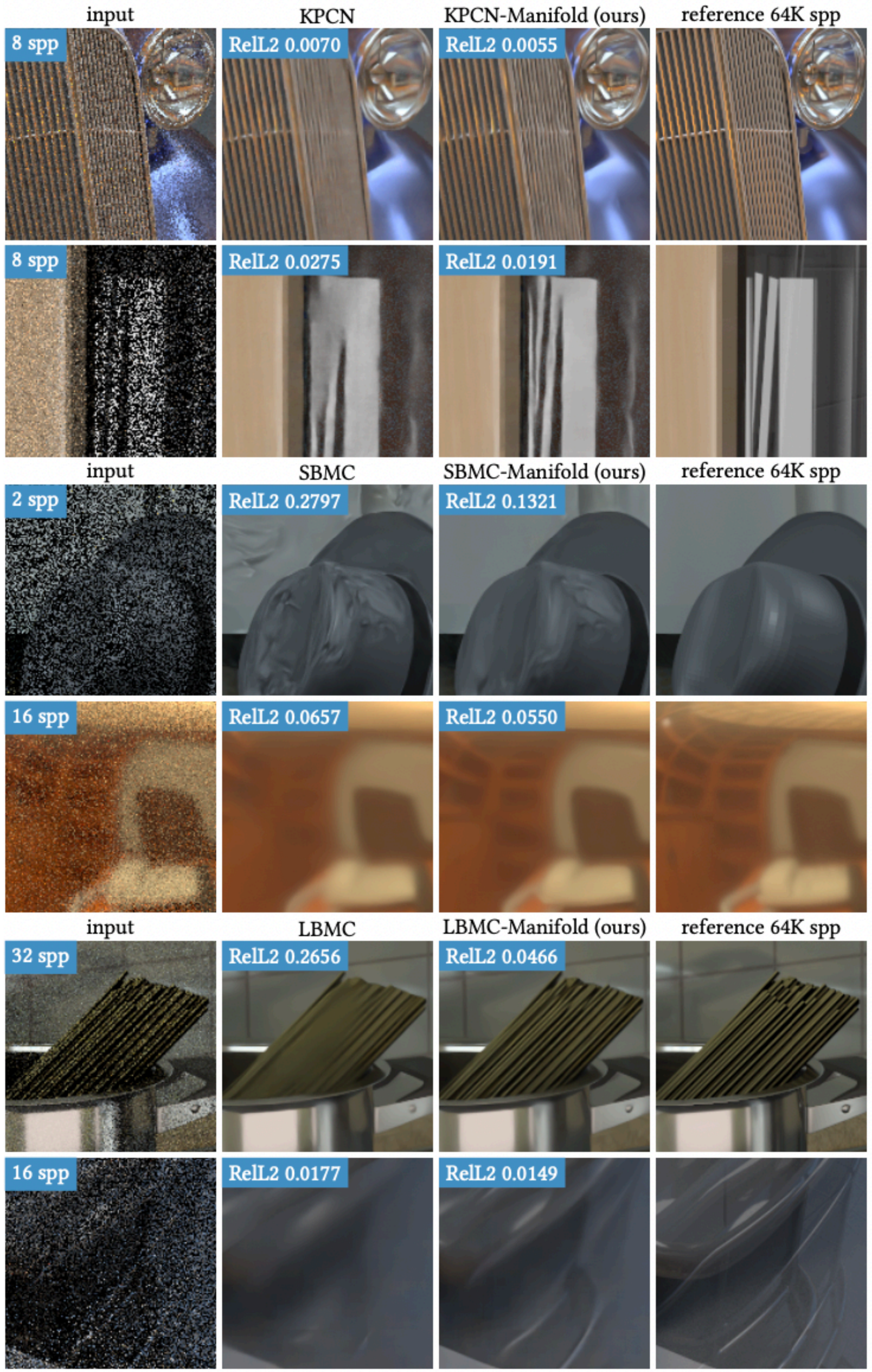
Experimental setup

- Use KPCN, SBMC, LBMC in comparing results(Vanilla models)
    - KPCN : Pixel based
    - SBMC, LBMC : Sample based
- Use a model that only minimize regression loss, and not the contrastive loss(Path models)
    - Demonstrate the effectiveness of manifold learning
- Use models that minimize both regression and contrastive loss(Manifold models)

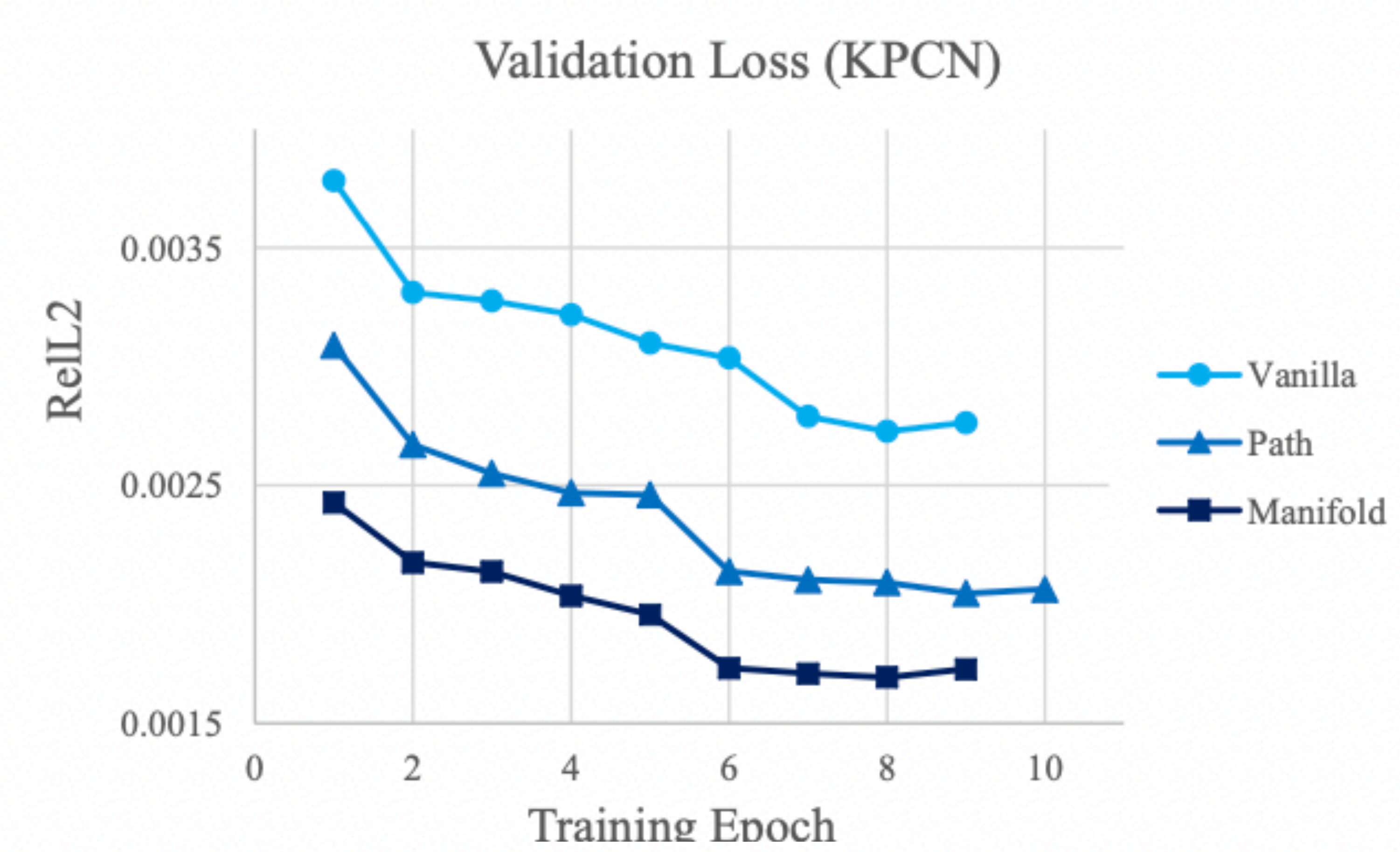
Results

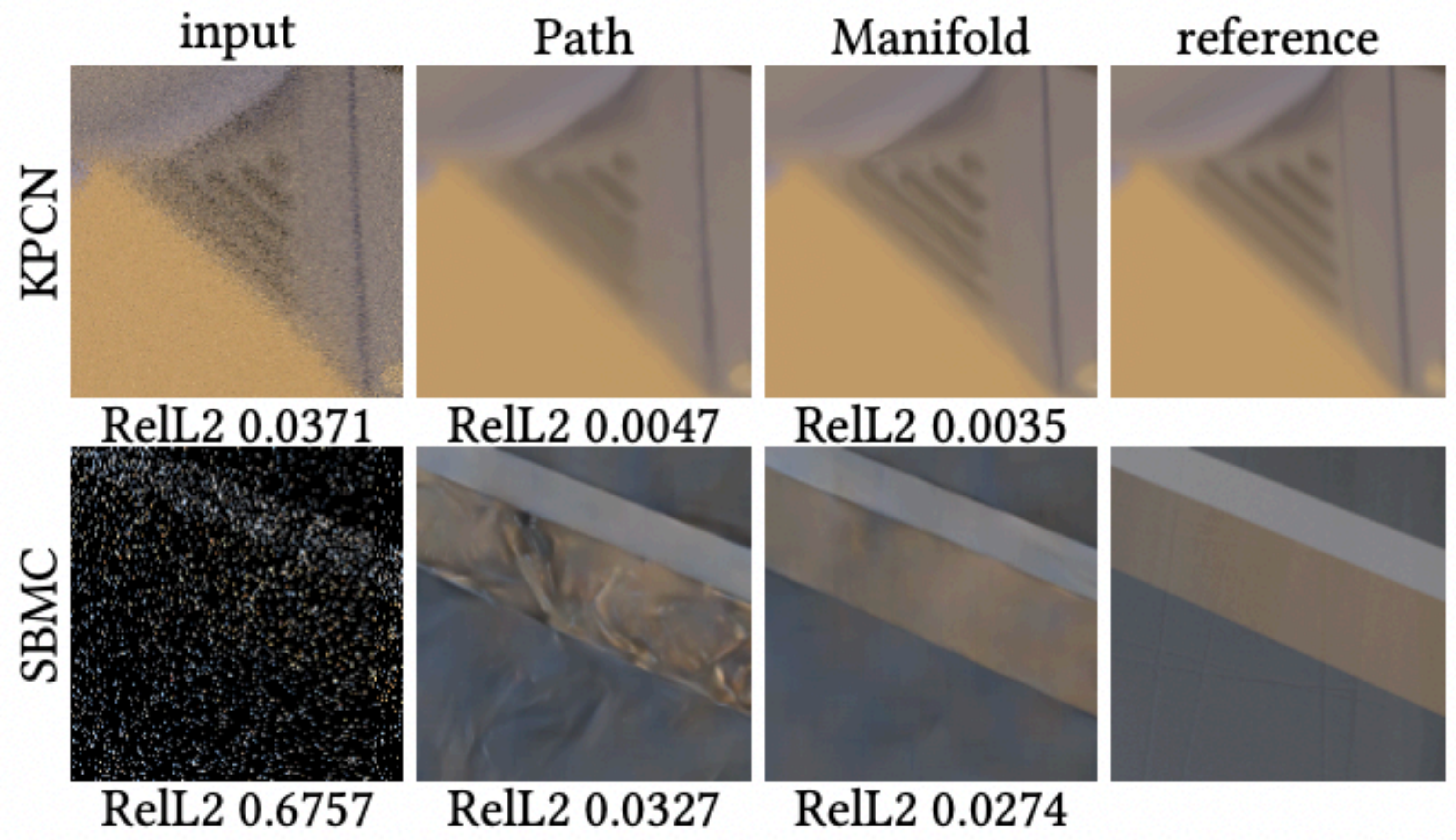


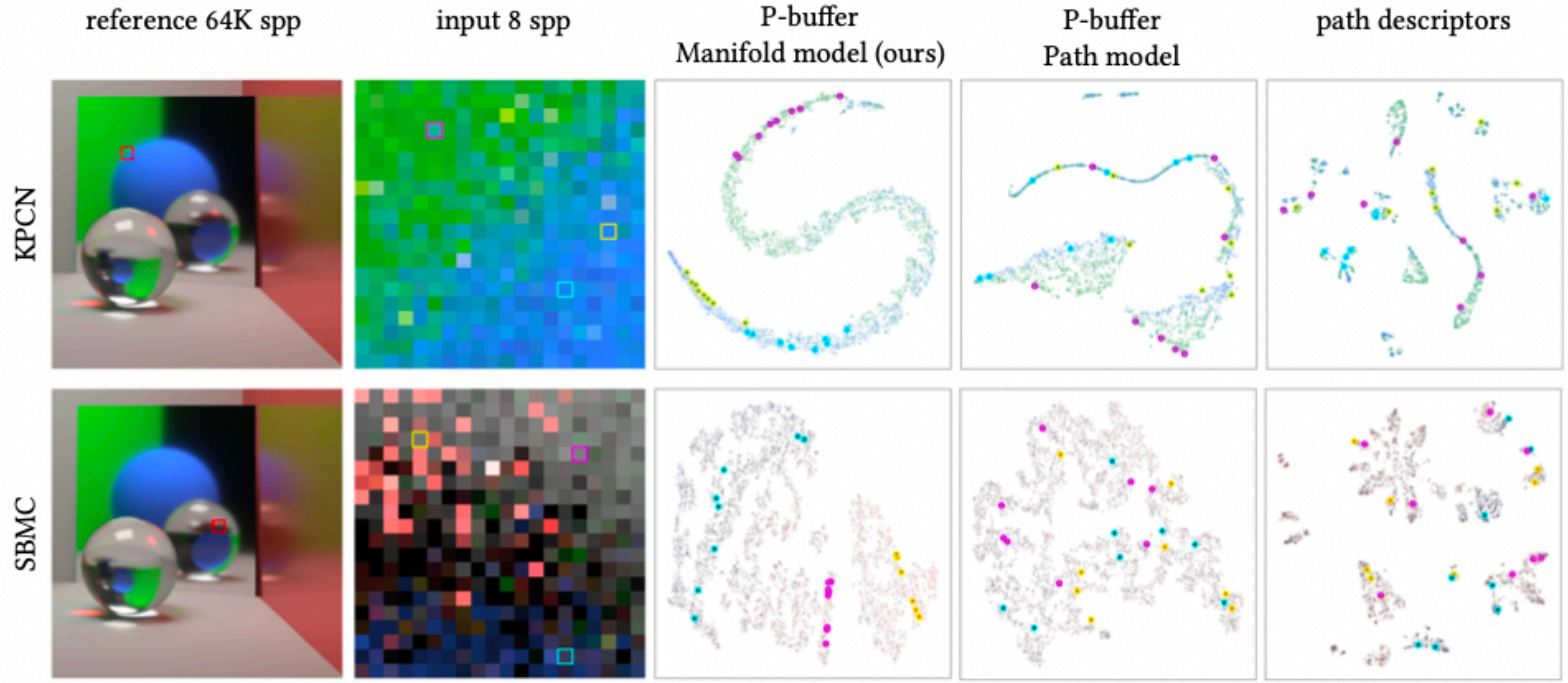
balancing parameter (λ)	0.01	0.1	0.5
RelL2 ($\times 10^{-3}$)	1.837	**1.693**	1.731
# of channels of P-buffer	3	6	12
RelL2 ($\times 10^{-3}$)	1.693	1.681	**1.644**



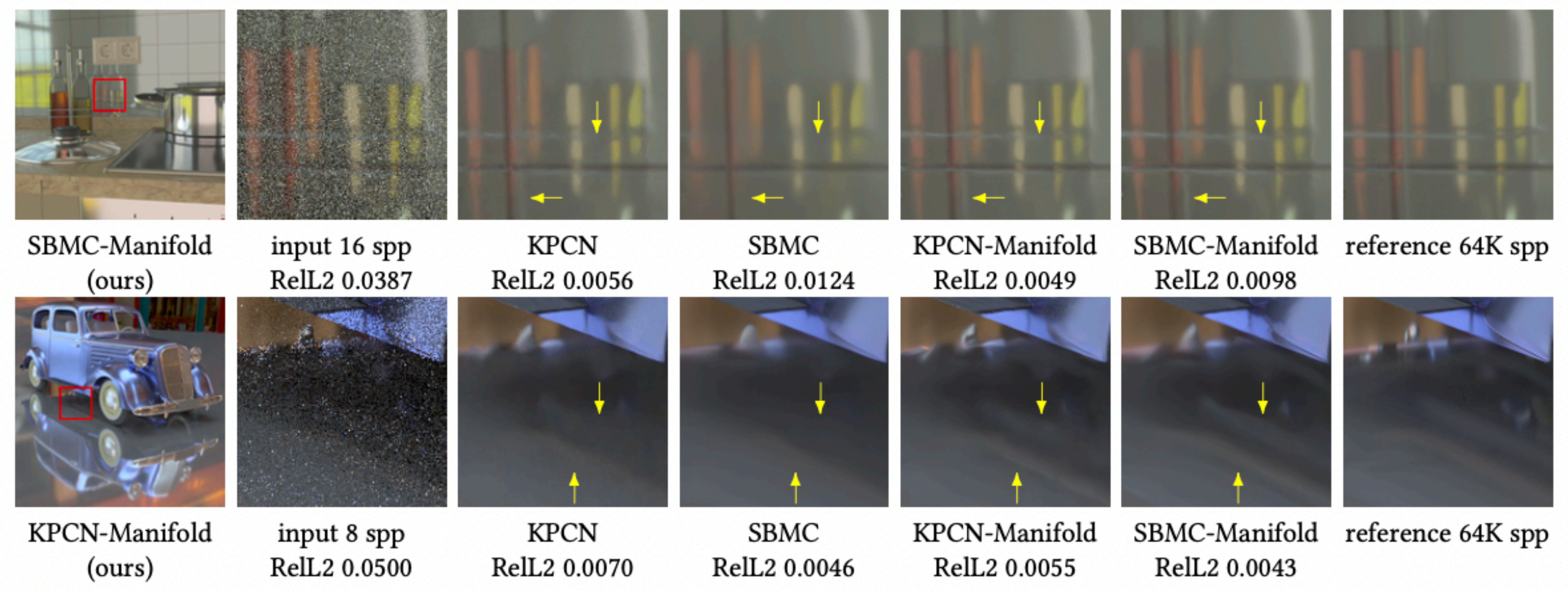
Results


Validation Loss (KPCN)
RelL2
0.0035
0.0025
0.0015
0
2
4
6
8
10
Training Epoch
Vanilla
Path
Manifold


input
Path
Manifold
reference
KPCN
RelL2 0.0371
RelL2 0.0047
RelL2 0.0035
SBMC
RelL2 0.6757
RelL2 0.0327
RelL2 0.0274


reference 64K spp
input 8 spp
P-buffer
Manifold model (ours)
P-buffer
Path model
path descriptors
KPCN
SBMC

Conclusion



SBMC-Manifold (ours)
input 16 spp RelL2 0.0387
KPCN RelL2 0.0056
SBMC RelL2 0.0124
KPCN-Manifold RelL2 0.0049
SBMC-Manifold RelL2 0.0098
reference 64K spp
KPCN-Manifold (ours)
input 8 spp RelL2 0.0500
KPCN RelL2 0.0070
SBMC RelL2 0.0046
KPCN-Manifold RelL2 0.0055
SBMC-Manifold RelL2 0.0043
reference 64K spp

Thank you!