THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學
Department of Computing
電子計算學系

oppo

ICCV
OCT 19-23, 2025
HONOLULU
HAWAII

Poster Session 6 & Exhibit Hall I #2441

Generalized and Efficient 2D Gaussian Splatting for Arbitrary-scale Super-Resolution

Presenter: Du Chen

Du Chen[1,2*], Liyi Chen[1*], Zhengqiang Zhang[1,2] and Lei Zhang[1,2]

[1]The Hong Kong Polytechnic University, [2]OPPO Research Institute

csdud.chen@connect.polyu.hk

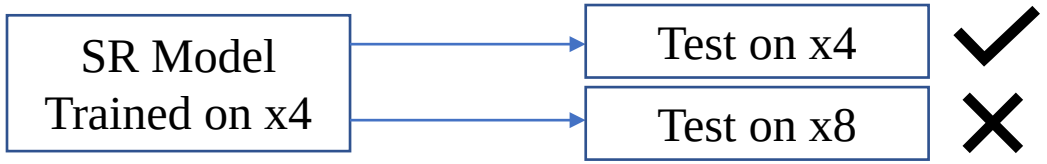
2025.10.23

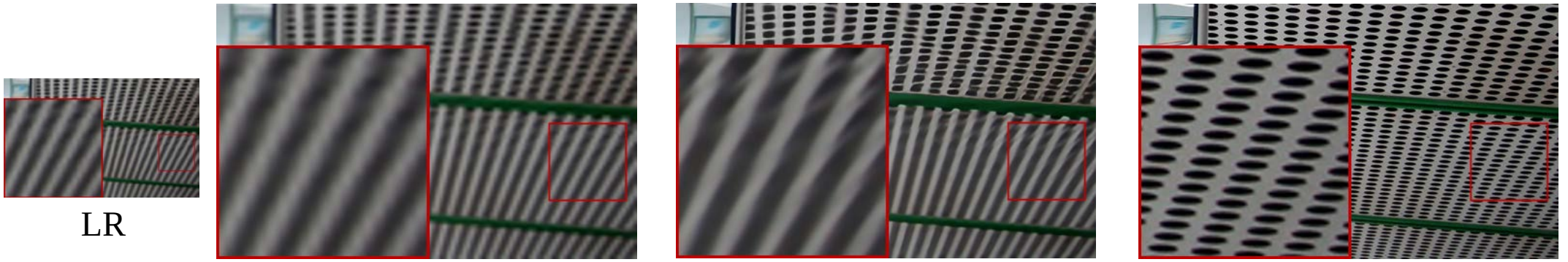
Observation

- Continuous super-resolution is a challenging task.
- Bicubic or bilinear interpolation will scarify image fidelity.

Bicubic function ≠ Complex non-linear mapping

- Fixed de-convolution or pixel-shuffle layers only operate on discrete pixels. Thus the scale is fixed and can only handle integer scaling factor.





LR | Bicubic | RDN [1] | GT

Introduction
Method
Results

Observation

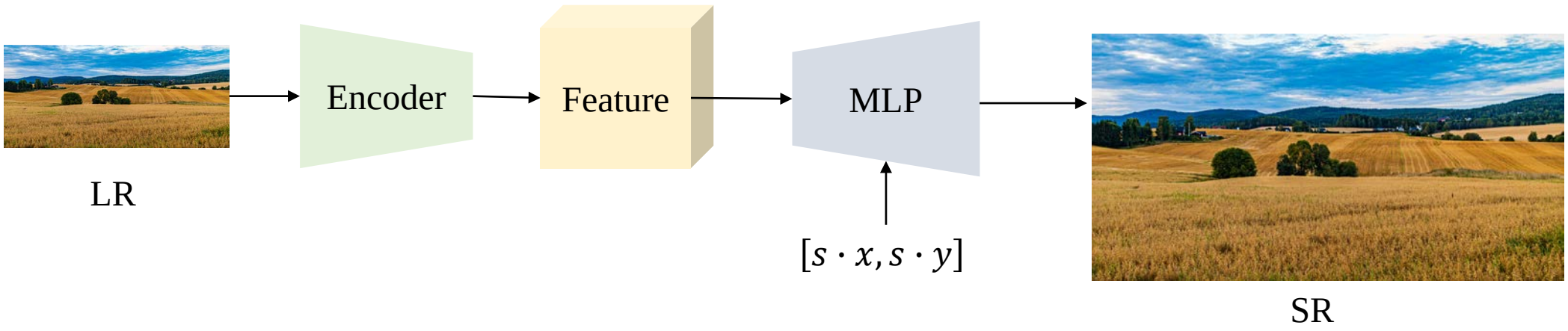
Implicit Neural Functions

- LIIF [1]
- Encode image into deep feature

$$F = E(I_{LR})$$

- Take the feature as input, and give extra position information as condition

$$I_{SR} = \theta([s \cdot x, s \cdot y], F_{x,y})$$



[1] Chen, Y., Liu, S., & Wang, X. “Learning continuous image representation with local implicit image function”, in CVPR 2021.

Observation

Implicit Neural Representations have the following limitations

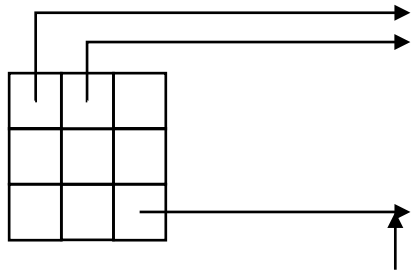
- Query the MLP numerous times. Inefficient.
- Poor representation capability in MLP.

$$F = E(I_{LR})$$

$$I_{SR} = \theta([s \cdot x, s \cdot y], F_{x,y})$$

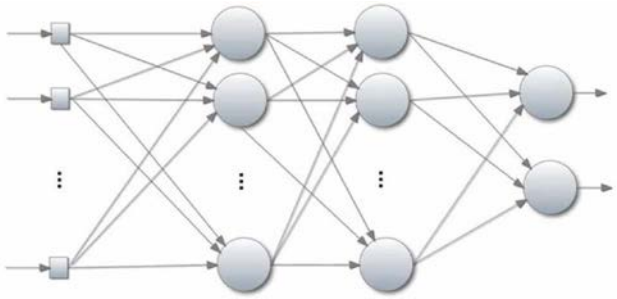
LR

Encoder



Deep feature

Pos(x,y)

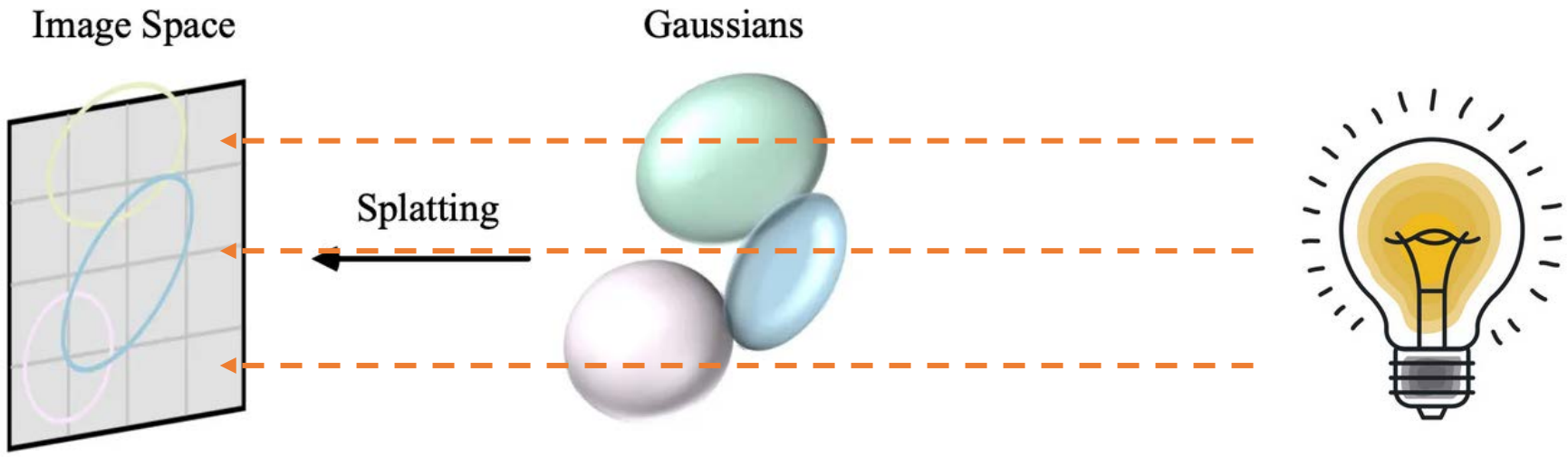


INR

SR

[1] Chen, Y., Liu, S., & Wang, X. “Learning continuous image representation with local implicit image function”, in CVPR 2021.

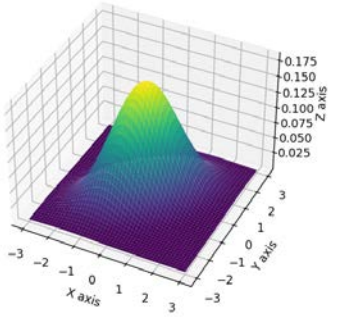
Gaussian Splatting [1]



The advantages of GS over INR

- Continuous representation, stronger representation ability than INR.
- Explicit representation and improved interpretability.
- Physically meaningful parameter.
- Efficient.

2D Gaussian Splatting



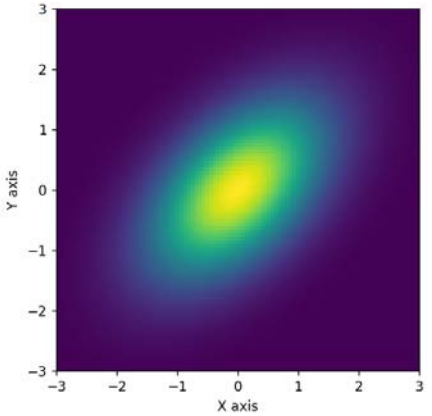
How to use Gaussians to represent an image

- We have a series of Gaussians:

$$G_1, G_2, G_3, \dots, G_n$$

- Each Gaussian G_i has 9 properties:

$$\sigma_x, \sigma_y, \rho_{xy}, \mu_x, \mu_y, \alpha, C \in [0,1]^3$$



σ_x, σ_y: Standard deviation on axis X, Y
ρ_{xy}: Correlation coefficient
μ_x, μ_y: Position parameter
α: Opacity, decide each Gaussian's weight
C : Normalized Peak RGB values

- 2D Gaussian function is depicted as:

$$f(x,y) = (2\pi\sigma_x\sigma_y\sqrt{1-\rho_{xy}^2})^{-1}\exp[-\frac{1}{2(1-\rho_{xy}^2)}] \times (\frac{\Delta x^2}{\sigma_x^2} - \frac{2\rho_{xy}\Delta x\Delta y}{\sigma_x\sigma_y} + \frac{\Delta y^2}{\sigma_y^2})]$$

2D Gaussian Splatting

How to use Gaussians to represent an image

- 2D Gaussian function is depicted as:

$$f(x,y) = (2\pi\sigma_x\sigma_y\sqrt{1-\rho_{xy}^2})^{-1}\exp[-\frac{1}{2(1-\rho_{xy}^2)}] \times (\frac{\Delta x^2}{\sigma_x^2} - \frac{2\rho_{xy}\Delta x\Delta y}{\sigma_x\sigma_y} + \frac{\Delta y^2}{\sigma_y^2})]$$

- where

$$\Delta x = x - \mu_x, \Delta y = y - u_y$$

- We define the RGB color at position (x, y) contributed by Gaussian G as:

$$\mathrm{G}(\mathrm{x},\mathrm{y}) = \alpha \cdot C \cdot f(x,y)$$

- The pixel color of the LR image at position (x, y) is represented as the summation of all N Gaussians :

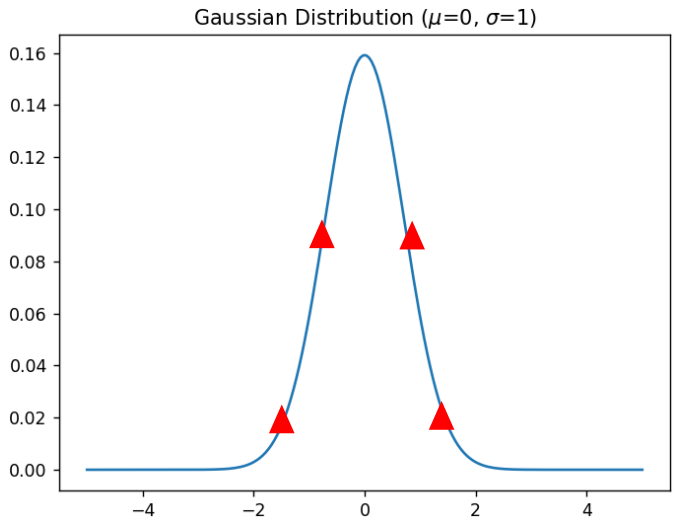
$$I_{LR}(x,y) = \sum_{i=1}^{N} G_{\mathrm{i}}(x,y)$$

- The SR image with scaling-factor s could be expressed as:

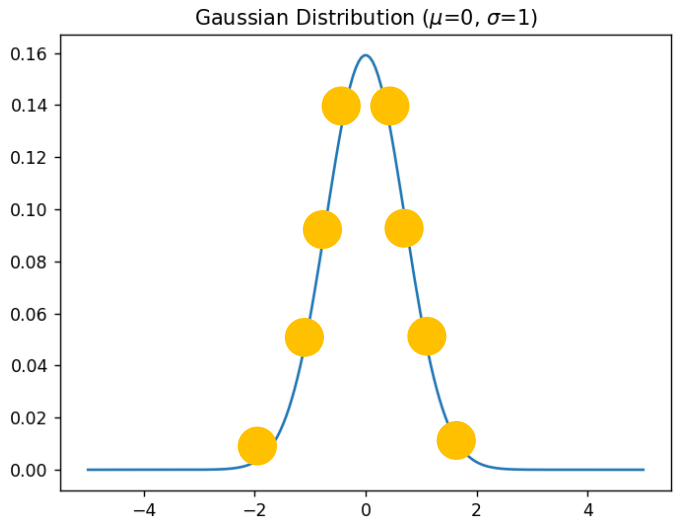
$$I_{SR}(x,y;s) = \sum_{i=1}^{N} G_{\mathrm{i}}(\frac{x}{s},\frac{y}{s})$$

Rendering/rasterization

Key point for Arbitrary-scale Super-resolution

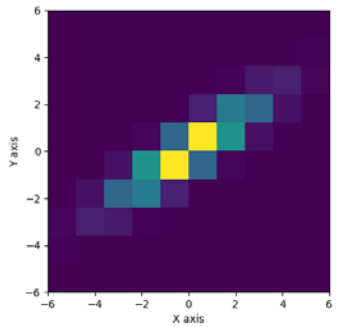


Scale x2

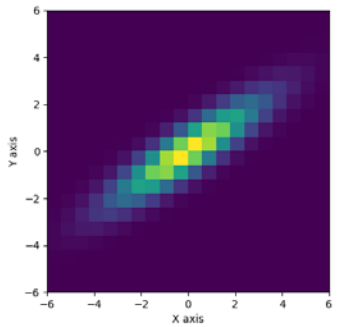


Scale x4

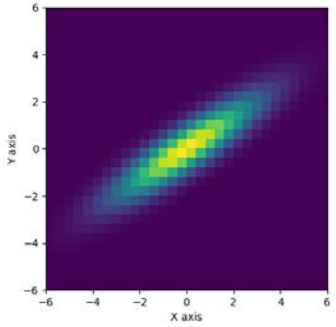
- Gaussian functions are infinity and continuous on 2D space, if we want to super-resolve an image with a scaling-factor, we just need to sample the corresponding pixels from the Gaussian distribution.
- The larger scaling factor, the smaller sampling step.



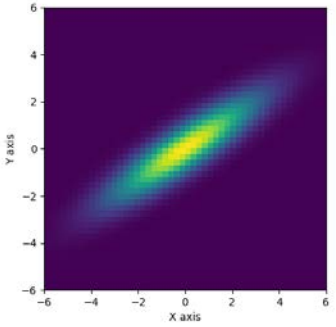
x2



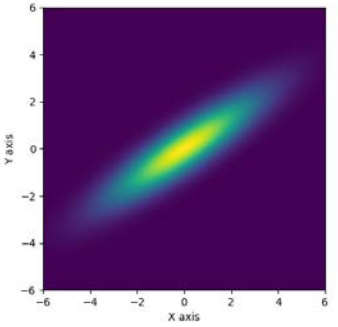
x4



x6

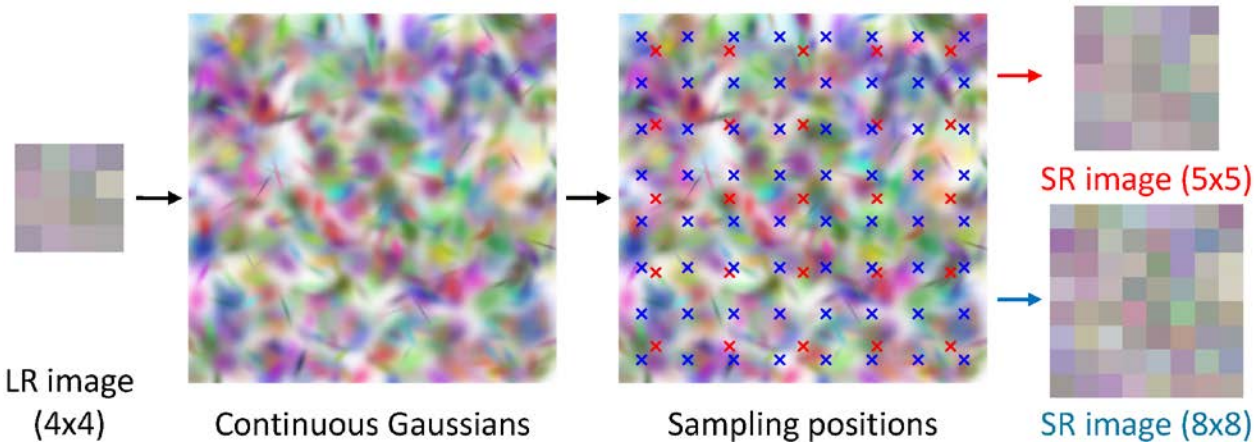


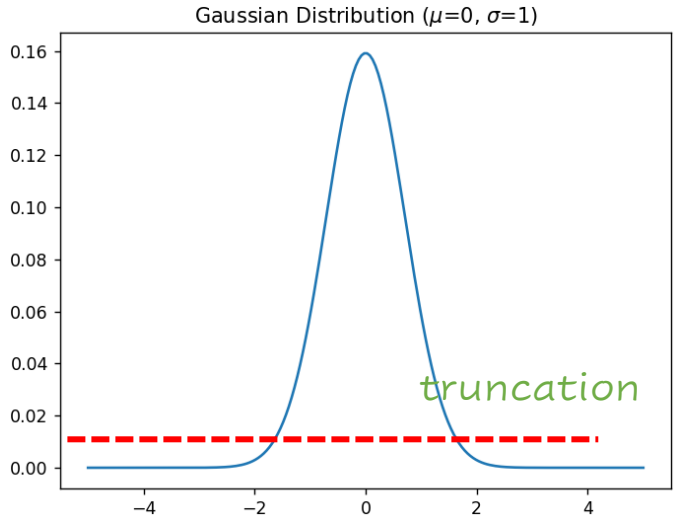
x10



x40

Fast Rasterization





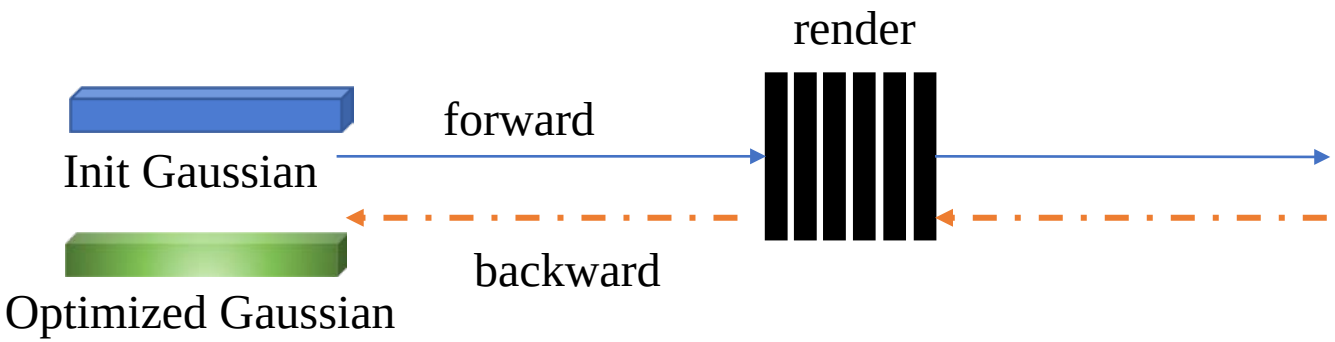
- Simply rendering a pixel by querying each pixel from all 2D Gaussians leads to a complexity of $O(s^2HWN)$.
- A Gaussian generally focuses on a local area and its contribution to pixel values decays rapidly with the increase of distance.
- We introduce a rasterization ratio r ≤ 1 to control the rendering range of each Gaussian.
- If the distance between the pixel and Gaussian center > r, then we will not use the Gaussian to render that pixel.
- Greatly reducing the computational complexity to $O(s^2r^2HWN)$.

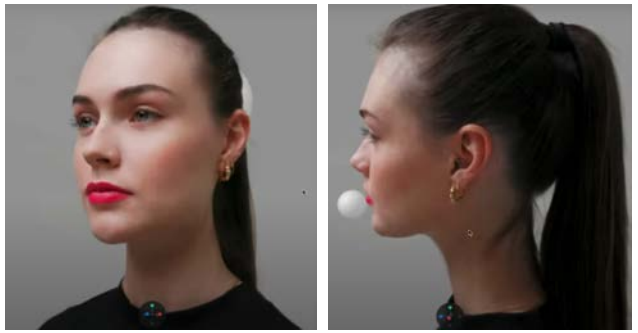
Algorithm 1: 2D GS Scale-aware Rasterization

Input: N 2D Gaussians $\{G_1, G_2, \ldots, G_N\}$; LR image of size (H, W); scale factor s; rasterization ratio r.
Output: Rendered image I_{SR}

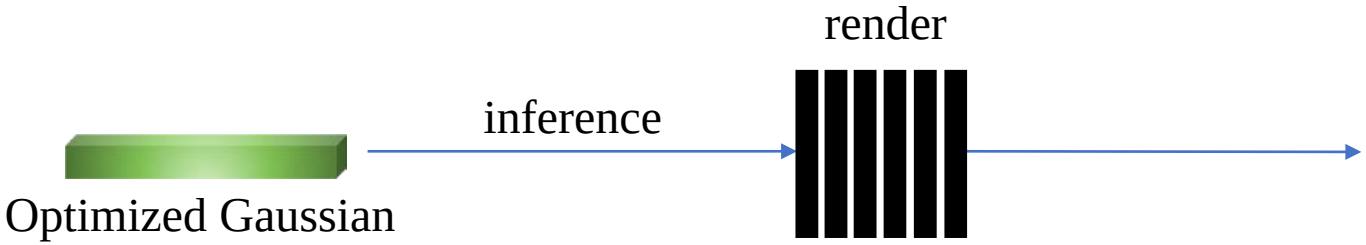
Initialize I_{SR} as an $(sH, sW, 3)$ array of zeros.
For *each* G_i *in* $\{G_1, G_2, \ldots, G_N\}$ **do**
  Initialize $\alpha, \mu_x, \mu_y, \sigma_x, \sigma_y, \rho, c$ from G_i.
  For *each pixel* (x, y) *in* I_{SR} **do**
    If $|x - \mu_x| < rsH$ *and* $|y - \mu_y| < rsW$ **then**
      Obtain $f(x/s, y/s)$ using Eq. 2;
      Obtain $G_i(x/s, y/s)$ using Eq. 1;
      $I_{SR}(x, y; s)$ += $G_i(x/s, y/s)$.
    end
  end
end

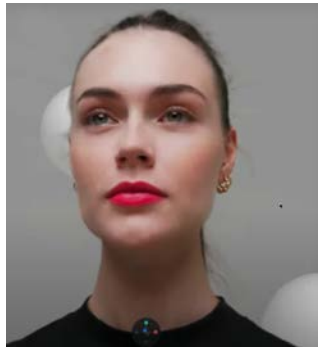
Existing 3D GS are just overfitting on a fixed scene





Images from two different cameras in a same scene



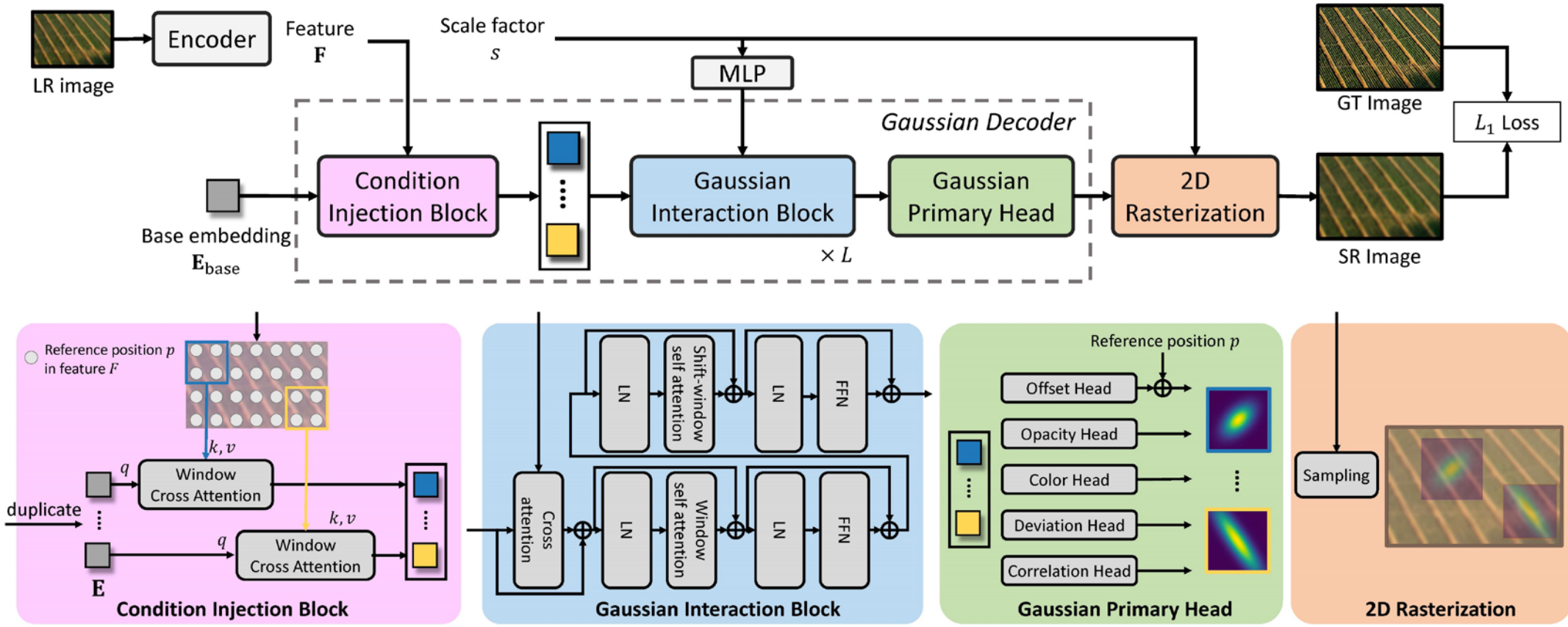


Synthesize a new viewpoint

3D GS cannot be directly applied to 2D image super-resolution

- Optimization-based manner.
- Cannot generalize to unseen testing image.

Backbone



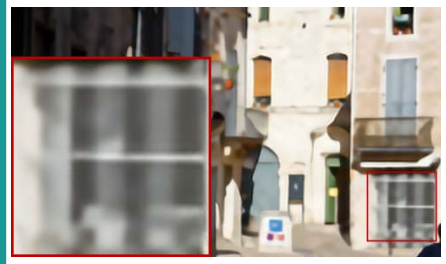
- Initial Gaussian embedding
- Condition injection block
- Gaussian interaction block
- Gaussian primary head
- Rasterization

Results

Scale	Metrics	Backbone: EDSR-baseline																	
		Testing Dataset: DIV2K									Testing Dataset: LSDIR								
		Meta-SR	LIIF	LTE	SRNO	LINF	LMF	Ciao-SR	Gaussian-SR	GSASR	Meta-SR	LIIF	LTE	SRNO	LINF	LMF	Ciao-SR	Gaussian-SR	GSASR
x2	PSNR	36.02	36.05	36.10	36.27	36.21	36.21	36.42	36.10	**36.65**	31.36	31.43	31.50	31.65	31.49	31.61	31.78	31.47	**32.14**
	SSIM	0.9455	0.9458	0.9461	0.9474	0.9461	0.9469	0.9476	0.9459	**0.9495**	0.9162	0.9170	0.9177	0.9197	0.9175	0.9193	0.9208	0.9174	**0.9251**
	LPIPS	0.0889	0.0879	0.0869	0.0833	0.0887	0.0832	0.0835	0.0888	**0.0767**	0.0984	0.0963	0.0947	0.0916	0.0973	0.0916	0.0890	0.097	**0.0823**
	DISTS	0.0571	0.0567	0.0567	0.0547	0.0564	0.0555	0.0543	0.0569	**0.0514**	0.0692	0.0688	0.0682	0.0662	0.0686	0.0667	0.0649	0.0685	**0.0612**
x4	PSNR	33.36	30.43	30.47	30.57	30.51	30.56	30.67	30.46	**30.89**	26.13	26.21	26.26	26.36	26.25	26.33	26.42	26.23	**26.65**
	SSIM	0.8367	0.8388	0.8395	0.8415	0.8396	0.8416	0.8431	0.8389	**0.8486**	0.7577	0.7614	0.7627	0.7666	0.7621	0.7656	0.7681	0.7615	**0.7774**
	LPIPS	0.2723	0.2662	0.2647	0.2616	0.2680	0.2607	0.2585	0.2684	**0.2518**	0.3074	0.2978	0.2957	0.2899	0.2998	0.2921	0.2865	0.3007	**0.2777**
	DISTS	0.1394	0.1403	0.1397	0.1384	0.1401	0.1379	0.1370	0.1406	**0.1301**	0.1666	0.1678	0.1664	0.1647	0.1675	0.1647	0.1631	0.1679	**0.1554**
x8	PSNR	26.72	26.87	26.93	27.00	26.91	26.97	27.04	26.76	**27.22**	23.23	23.32	23.37	23.43	23.34	23.40	23.47	23.27	**23.58**
	SSIM	0.7135	0.7207	0.7218	0.7243	0.7207	0.7235	0.7256	0.7155	**0.7321**	0.6032	0.6123	0.6137	0.6171	0.6118	0.6156	0.6194	0.6059	**0.6269**
	LPIPS	0.4365	0.4212	0.4321	0.4261	0.4285	0.4264	0.4169	0.4445	**0.4077**	0.5020	0.4812	0.4939	0.4848	0.4868	0.4883	0.4703	0.5068	**0.4611**
	DISTS	0.2305	0.2337	0.2348	0.2330	0.2341	0.2328	0.2314	0.2386	**0.2214**	0.2643	0.2673	0.2671	0.2640	0.2677	0.2654	0.2619	0.2713	**0.2518**
x16	PSNR	24.00	24.13	24.20	24.25	24.14	24.23	24.27	23.80	**24.38**	21.22	21.29	21.34	21.38	21.31	21.32	21.40	21.12	**21.42**
	SSIM	0.6307	0.6402	0.6407	0.6423	0.6394	0.6418	0.6443	0.6306	**0.6473**	0.5120	0.5221	0.5227	0.5245	0.5209	0.5213	0.5271	0.5122	**0.5296**
	LPIPS	0.5924	0.5754	0.5912	0.5847	0.5876	0.5861	0.5666	0.6325	**0.5563**	0.6687	0.6529	0.6691	0.6617	0.6622	0.6738	0.6386	0.7101	**0.6280**
	DISTS	0.3297	0.3357	0.3398	0.3367	0.3412	0.3371	0.3336	0.3582	**0.3242**	0.3638	0.3683	0.3701	0.3671	0.3727	0.3687	0.3636	0.3866	**0.3538**
x24	PSNR	22.63	22.74	22.81	22.84	22.74	22.83	22.87	22.41	**22.90**	20.25	20.31	20.35	20.39	20.32	20.34	**20.40**	20.11	20.38
	SSIM	0.6027	0.6108	0.6112	0.6119	0.6101	0.6117	0.6141	0.6037	**0.6150**	0.4854	0.4938	0.4938	0.4845	0.4930	0.4924	0.4967	0.4862	**0.4973**
	LPIPS	0.6512	0.6444	0.6600	0.6539	0.6545	0.6561	0.6352	0.7042	**0.6299**	0.7234	0.7214	0.7374	0.7292	0.7261	0.7421	0.7067	0.7812	**0.7028**
	DISTS	0.4022	0.4009	0.4063	0.4026	0.4106	0.4036	0.3980	0.4380	**0.3877**	0.4292	0.4274	0.4308	0.4269	0.4348	0.4297	0.4222	0.4579	**0.4117**
x30	PSNR	21.97	22.07	22.12	22.16	22.07	22.03	22.18	21.75	**22.19**	19.75	19.80	19.84	19.86	19.81	19.83	**19.87**	19.60	19.82
	SSIM	0.5982	0.5998	0.6000	0.6003	0.5994	0.5966	0.6021	0.5943	**0.6025**	0.4754	0.4824	0.4823	0.4824	0.4817	0.4818	0.4839	0.4766	**0.4842**
	LPIPS	0.7018	0.6773	0.6914	0.6862	0.6859	0.7137	0.6682	0.7317	**0.6648**	0.7386	0.7537	0.7676	0.7610	0.7575	0.7645	**0.7284**	0.8084	0.7368
	DISTS	0.4474	0.4375	0.4445	0.4400	0.4500	0.4476	0.4339	0.4821	**0.4232**	0.4702	0.4603	0.4645	0.4597	0.4702	0.4608	0.4535	0.4986	**0.4417**

Results

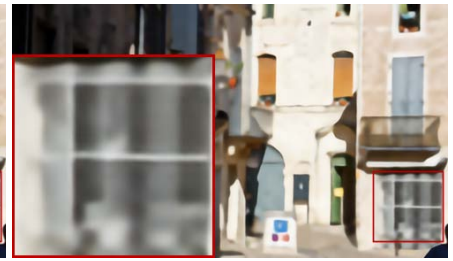
Introduction
Method
Results

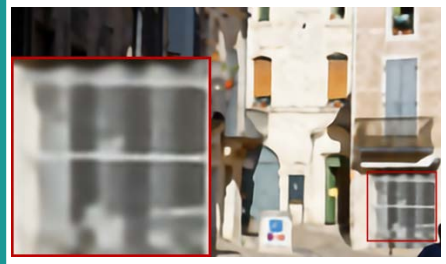

Meta-SR

LIIF

LTE

SRNO


LINF


LMF

CiaoSR

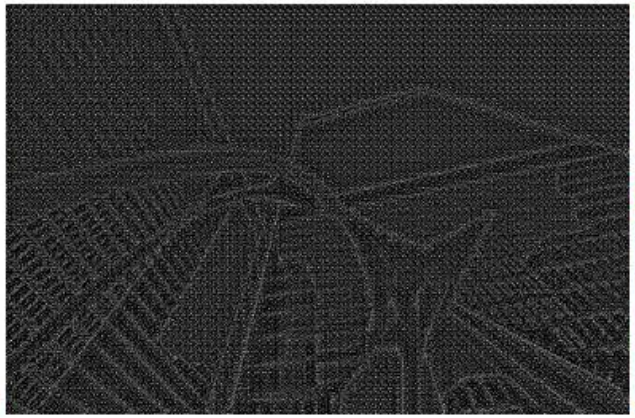
GaussianSR

GSASR (Ours)

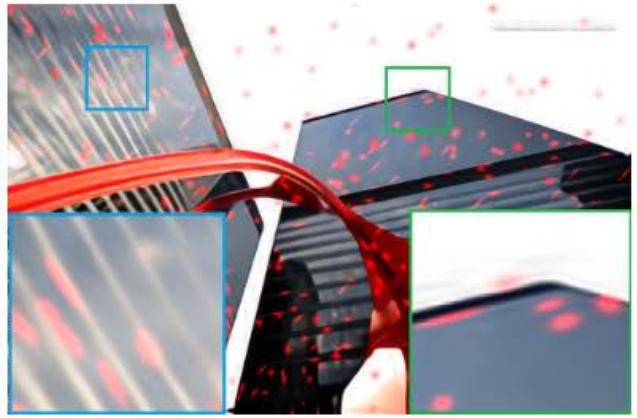
GT

Position and shape of Gaussians

SR Image



Position Distribution of Gaussians



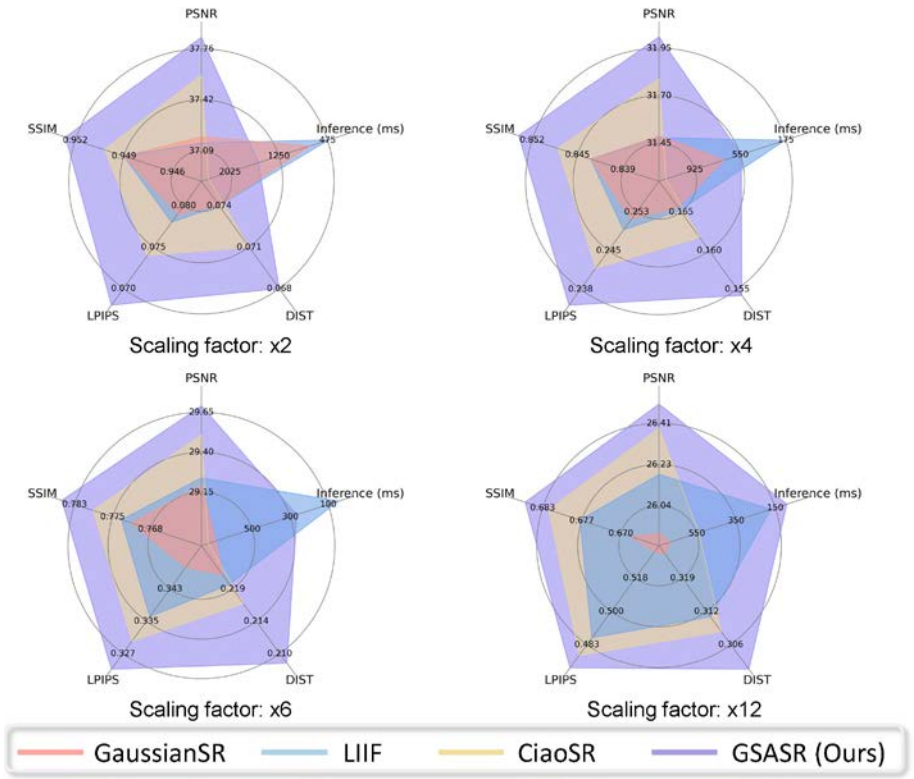
Colorized Gaussians

- Gaussians are evenly distributed on flatten areas, while focus on complex textures.
- Gaussians will learn to fit the direction and shape of edge textures.
- Gaussian shows powerful representation capability.

Results

Introduction
Method
Results

Scale	Computational Cost and Performance	Backbone: EDSR-baseline Testing GT Size: 720 * 720								
		Meta-SR	LIIF	LTE	SRNO	LINF	LMF	CiaoSR	GaussianSR	GSASR
x2	PSNR	37.09	37.14	37.16	37.34	37.42	37.29	37.59	37.18	**37.84**
	SSIM	0.9475	0.9487	0.9491	0.9501	0.9490	0.9498	0.9502	0.9489	**0.9522**
	LPIPS	0.0787	0.0777	0.0768	0.0743	0.0785	0.0736	0.0744	0.0785	**0.0676**
	DISTS	0.0741	0.0737	0.0736	0.0713	0.0731	0.0724	0.0710	0.0738	**0.0676**
	Inference Time	186	454	126	107	**86**	209	23603	754	1573
	GPU Memory	670	548	**490**	6301	3573	4005	49152	5200	13367
x4	PSNR	31.38	31.48	31.48	31.65	31.62	31.60	31.79	31.49	**32.01**
	SSIM	0.8417	0.8437	0.8444	0.8472	0.8444	0.8466	0.8482	0.8437	**0.8536**
	LPIPS	0.2545	0.2490	0.2472	0.2429	0.2506	0.2438	0.2416	0.2511	**0.2344**
	DISTS	0.1638	0.1652	0.1646	0.1626	0.1653	0.1930	0.1617	0.1656	**0.1542**
	Inference Time	**47**	182	114	95	66	68	1165	686	543
	GPU Memory	432	308	**279**	6275	3357	1139	3331	5048	3420
x6	PSNR	29.08	29.24	29.25	29.39	29.33	29.34	29.51	29.19	**29.69**
	SSIM	0.7686	0.7735	0.7744	0.7777	0.7738	0.7766	0.7790	0.7715	**0.7851**
	LPIPS	0.3455	0.3362	0.3420	0.3361	0.3411	0.3374	0.3298	0.3492	**0.3223**
	DISTS	0.2179	0.2199	0.2195	0.2176	0.2201	0.2179	0.2171	0.2214	**0.2083**
	Inference Time	**46**	176	118	94	65	55	716	692	265
	GPU Memory	387	264	**239**	6270	3316	912	1548	5224	1578
x8	PSNR	27.74	27.89	27.94	28.04	27.97	28.00	28.14	27.77	**28.25**
	SSIM	0.7216	0.7284	0.7297	0.7326	0.7285	0.7314	0.7345	0.7235	**0.7397**
	LPIPS	0.4132	0.3956	0.4062	0.3988	0.4025	0.4017	0.3892	0.4187	**0.3810**
	DISTS	0.2543	0.2569	0.2573	0.2555	0.2576	0.2570	0.2542	0.2623	**0.2448**
	Inference Time	**42**	170	112	96	64	64	616	666	195
	GPU Memory	371	248	**224**	6269	3302	832	1503	5012	1051
x12	PSNR	26.03	26.18	26.23	26.32	26.24	26.28	26.39	25.91	**26.50**
	SSIM	0.6687	0.6773	0.6783	0.6805	0.6768	0.6796	0.6825	0.6688	**0.6864**
	LPIPS	0.4990	0.4861	0.5011	0.4941	0.4958	0.4959	0.4769	0.5314	**0.4701**
	DISTS	0.3063	0.3114	0.3141	0.3113	0.3140	0.3127	0.3084	0.3241	**0.3007**
	Inference Time	**41**	172	113	91	64	52	540	688	91
	GPU Memory	360	237	**214**	6268	3292	775	1470	5214	472



Thank you!