دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش پروژه پایانی درس شبکه عصبی

نگارش

غلامرضا دار

استاد درس

دکتر رضا صفابخش

تابستان ۱۴۰۱

چکیده

در این گزارش به آشنایی با یکی از روش‌های جدید برای بازنمایی صحنه و تولید تصاویر بدیع از نماهای مختلف به نام میدان‌های تابندگی عصبی میپردازیم. مشکلات و ضعف های ین روش را بیان می‌کنیم و راه‌حل‌های ارائه شده برای تعدادی از مشکلات آن را مورد بررسی قرار می‌دهیم.

کلمات کلیدی:

شبکه‌های عصبی عمیق، پرسپترون چندلایه، بازنمایی سه‌بعدی صحنه

فهرست مطالب

فهرست اشکال

فهرست جداول

۱ . مقدمه

در سال‌های اخیر و با پیش‌رفت شبکه‌های عصبی و معرفی معماری‌های جدید، مشاهده شده است که در بسیاری از زمینه‌های پژوهشی، از این نوع شبکه‌ها برای حل انواع مختلف مسائل استفاده می‌شود. یکی از زمینه‌های پرکاربرد و فعال در علم کامپیوتر، گرافیک کامپیوتری است. محققان فعال در گرافیک کامپیوتری همواره سعی دارند مشکلات متعددی که در این زمینه وجود دارد را به کمک روش‌های مختلف حل کنند. در بسیاری از مسائل این شبکه‌های عصبی هستند که بعضاً با ترکیب با روش‌های قدیمی، نتایج خارق‌العاده ای کسب می‌کنند و بهبود دادن این شبکه‌ها در جهت رسیدن به نتایج بهتر همیشه جزء جدایی‌ناپذیر موضوعات تحقیقاتی در این زمینه است.

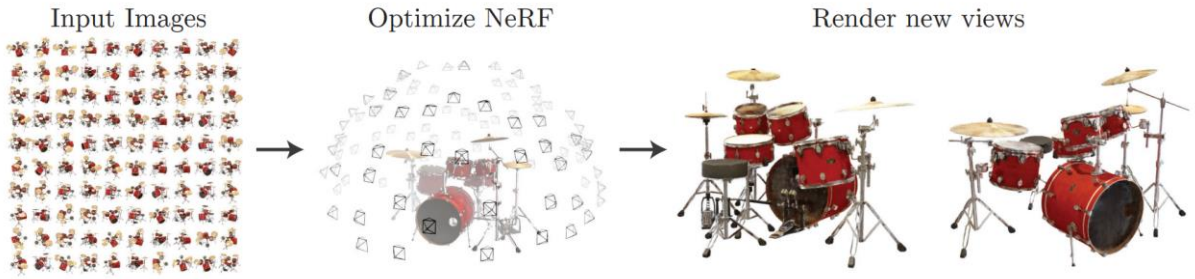
از جمله مسائل بسیار پرکاربرد و سختی که در زمینه گرافیک کامپیوتری وجود دارد، مسئله **سنتز نما**[1] نام دارد. هدف این مسئله تولید تصاویری از نماها یا زوایای‌دید جدید از یک صحنه‌ی به‌خصوص، با دیدن تعداد محدودی تصویر از آن صحنه است. وجود این نماهای جدید کاربردهای فراوانی در زمینه‌های مختلف مانند آموزش اتومبیل‌های خودران، ثبت و بازنمایش آثار و اماکن باستانی، و ... دارد.

در این گزارش، به بررسی جزئی سابقه روش‌های موجود برای حل این مسئله میپردازیم، یکی از روش‌های موفق اخیر را معرفی و تحلیل میکنیم و پس از بیان مشکلاتی که هنوز در این زمینه وجود دارند، تعدادی از مقالات اخیر که این مشکلات را تا حد زیادی بهبود می‌دهند را بررسی میکنیم.

[1] View Synthesis

۲. میدان‌های تابندگی عصبی(NeRF)[۲]

این روش ابتدا توسط Mildenhall و همکاران در [۱] معرفی شد. ایده اصلی آنها برای تولید نماهای جدید این بود که ابتدا با استفاده از تصاویر موجود، یک بازنمایی از تمام صحنه موجود در تصویر را در قالب یک میدان پیوسته تولید کنند، سپس، با کمک روش‌های موجود در زمینه ترسیم تصاویر از حجم[۳]، تصاویر جدید را از نماهای مختلف تولید کنند.



شکل ۱ ـ تولید نماهای جدید به کمک NeRF

در شکل ۱ می‌توانید به طور کلی عملکرد این روش را مشاهده کنید. تعداد ۲۰ الی ۳۰ تصویر، از زوایای مختلف از یک صحنه خاص جمع آوری شده اند. این تصاویر به عنوان ورودی به شبکه داده می‌شوند و شبکه در مرحله آموزش سعی می‌کند یک بازنمایی از صحنه‌ی مشترک موجود در تصاویر را در وزن‌های خود بی‌آموزد. در مرحله اجرا می‌توان حجم ذخیره شده در این شبکه را با استفاده از روش‌های ترسیم کامپیوتری تصاویر از حجم، به تصاویر متعدد از نماهای مختلف تبدیل کرد.

جزئیات هرکدام از مراحل ذکر شده را در بخش‌های بعد تحلیل و بررسی خواهیم کرد.

[2] Neural Radiance Fields

[3] Volume Rendering

۱ـ۲. کارهای پیشین

یک دسته از روش‌ها برای سنتز نما، از ساختار مشبکی از رئوس و یال‌های اجسام سه بعدی به نام Mesh استفاده می‌کنند. این روش‌ها، در ادامه با استفاده از صفحه‌کننده‌ها[۴] [۹] یا ترسیم‌کننده‌های اشعه[۵] [۱۰] که از روش‌های کلاسیک و پرکاربرد تولید تصاویر از اجسام سه بعدی هستند،از Mesh های موجود تصاویری از نماهای مختلف تولید می‌کنند. از بزرگترین مشکلات این روش‌ها، وابستگی آنها به وجود یک Mesh نمونه برای شروع فرآیند بهینه سازی است. در بسیاری از کاربردهای دنیای واقعی، این Mesh ها وجود ندارند و این باعث می‌شود این روش‌ها از محبوبیت کمتری برخوردار باشند. اما باید توجه کرد که برای مسائلی که در آنها به Mesh نمونه دسترسی داریم، این روش ها خروجی بسیار مناسبی تولید می‌کنند زیرا صفحه‌کننده‌ها و ترسیم‌کننده‌های اشعه امروزی بسیار قوی هستند و قادرند نتایج نزدیک به واقعیت ارائه دهند.

دسته دیگری از روش‌ها، بر پایه بازنمایی صحنه به شکل یک حجم عمل می‌کنند. این روش‌ها عموماً دارای نقص[۶]های کمتری در خروجی نسبت به روش‌های بر اساس Mesh هستند. تعدادی از روش‌های این دسته، صحنه را به تعدادی پیکسل حجمی[۷] یا Voxelجدول‌بندی میکنند سپس با استفاده از تصاویر ورودی، سعی می‌کنند این Voxel ها را رنگ آمیزی کنند[۱۱]. کیفیت این نوع روش ها وابسته به اندازه جدول بندی صحنه است. در همین دسته، روش‌های دیگری وجود دارند که با کمک شبکه‌های عصبی پیچشی[۸]، و اعمال آنها به همین Voxel ها سعی می‌کنند اثرات منفی ناشی از این گروه‌بندی را که معمولا در مرز بین Voxelها رخ می‌دهد را از بین ببرند.

روش ارائه شده در این این مقاله، تفاوت اساسی ای با روش‌های براساس Voxel دارد به این شکل که به جای تقسیم بندی صحنه به سلول های گسسته، یک بازنمایی پیوسته از صحنه را می‌آموزد. این امر باعث می‌شود نتایج تولیدی توسط این روش عموماً نرم‌تر باشد و همچنین به دلیل اینکه این روش صحنه را درون وزن‌های خود ذخیره میکند، نسبت به روش‌های بر پایه Voxel حجم ذخیره‌سازی بسیار کمتری نیاز دارد.

---

[4] Rasterizer

[5] Raytracer

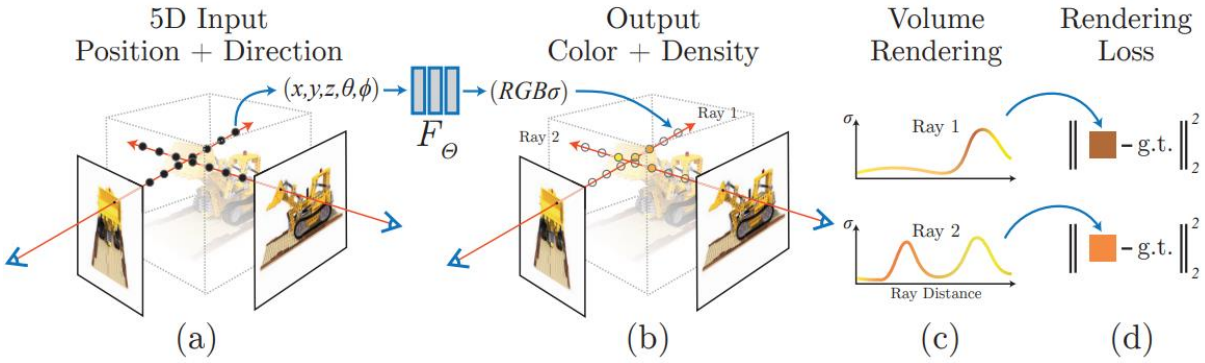
[6] Artifact

[7] Voxel(Volume Pixel)

[8] Convolutional Neural Networks

۲ـ۲. بازنمایی پیوسته صحنه توسط یک میدان 5D

همانطور که ذکر شد، این روش، صحنه موجود در تصاویر ورودی را به شکل یک تابع برداری پیوسته ذخیره می‌کند. تابع مذکور به شکل $F_\theta: (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$ است که در آن، $\mathbf{x} = (x, y, z)$ موقعیت مکانی یک نقطه از صحنه است و $\mathbf{d} = (\theta, \phi)$ زوایایی هستند که مشخص می‌کنند از چه نمایی به نقطه $\mathbf{x}$ نگاه می‌کنیم. خروجی های این تابع، $\boldsymbol{c} = (r, g, b)$ یا همان رنگ نقطه مذکور از زاویه دید $\mathbf{d}$ است و σ چگالی یا همان زخیم بودن نقطه است که برای اجسام شفاف عدد کوچکی است و برای اجسام غیرشفاف، عدد بزرگی است.



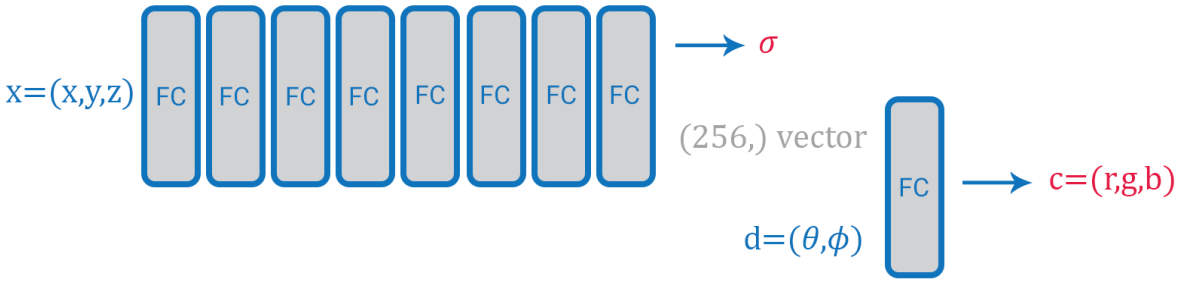
شکل ۲ ـ شمای کلی بازنمایی صحنه و تولید تصاویر جدید در روشNeRF

با داشتن توصیف کاملی از صحنه به شکل F_θ، قادر خواهیم بود با انتخاب هر $\mathbf{d_0}$ دلخواهی، یک تصویر از زاویه دید $\mathbf{d_0}$ تولید کنیم. نحوه تولید این تصویر، به این شکل است که تابع F_θ آموزش دیده شده را، با ورودی $\mathbf{d} = \mathbf{d_0}$ و $\mathbf{x}$ های مختلف صدا می‌زنیم و تصویر نهایی جدید تولید می‌شود. نحوه تولید رنگ $\mathbf{c} = (r, g, b)$ و چگالی σ در بخش ۲ـ۳ توضیح داده خواهد شد.

در این مقاله، تابع F_θ به کمک یک شبکه پرسپترون چندلایه یا MLP[9] مدل‌سازی شده است. در حین آموزش، وزن‌های این شبکه(θ)، آموزش می‌بینند و این شبکه پس از اتمام آموزش قادر به بیان صحنه‌ی موجود به کمک وزن‌های خود خواهد بود. اگر کمی توجه کنید، به این نکته پی خواهید برد که رنگ نقاط، علاوه بر موقعیت نقاط، به زاویه دید دوربین نیز بستگی دارد(به عنوان مثال به بخش c در شکل ۴ توجه کنید)، اما چگالی نقاط صرفا به موقعیت نقاط بستگی دارد(البته ممکن است اجناس[10] خاصی در طبیعت وجود داشته باشند که با توجه به زاویه دید دوربین، خواص مختلفی نشان دهند اما در این مقاله، جهت ساده سازی مسئله، از آن نوع اجناس صرف نظر شده است).

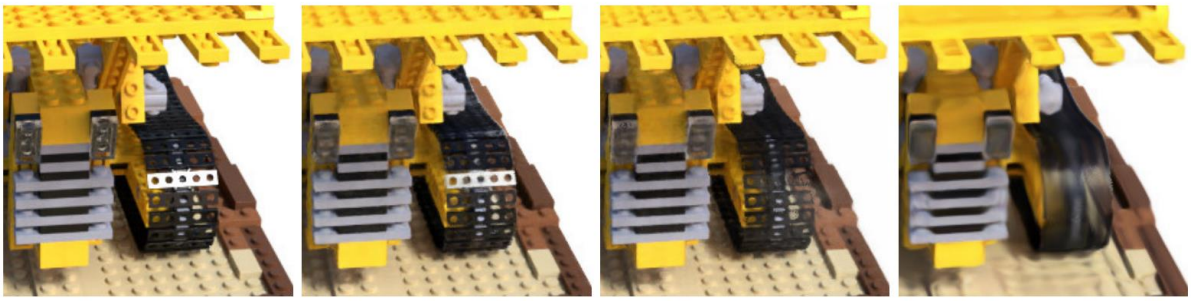
[9] Multilayer Perceptron

[10] Materials



شکل ۳ـ معماری شبکه مورد استفاده در NeRF

در این مقاله برای مقابله با این مشکل، شبکه را به دو زیرشبکه کوچکتر تقسیم می‌کنند. همانطور که در شکل ۳ مشاهده می‌کنید، ابتدا موقعیت فضایی نقاط، به عنوان ورودی به زیرشبکه اول، که دارای ۸ لایه تماماً متصل است، داده می‌شود. این ۸ لایه یک بازنمایی درونی از نقاط به ابعاد ۲۵۶ و خروجی σ را تولید می‌کنند. در ادامه این بازنمایی درونی به علاوه زاویه مورد نظر، به زیرشبکه دوم داده می‌شود که صرفا یک لایه تماماً متصل است و خروجی رنگ تولید می‌شود. با این کار، خروجی σ کاملا مستقل از زاویه دید دوربین تولید می‌شود. این کار مسئله را ساده‌تر می‌کند و باعث می‌شود تصاویر خروجی همانطور که در بخش c در شکل ۴ دیده می‌شود بهبود چشم‌گیری بی‌یابند.



شکل ٤ ـ مقایسه خروجی شبکه NeRF در حالت های مختلف

۳ـ۲. تولید تصاویر با کمک ترسیم تصاویر از حجم

در بخش قبل دیدیم تابع F_θ چگونه می‌تواند صحنه را در خود ذخیره کند. در این بخش می‌خواهیم ببینیم چگونه می‌توان با استفاده از روش‌های ترسیم تصاویر از حجم، تصاویر مختلفی را از زوایای مختلف تولید کرد. رابطه اصلی تولید تصاویر به این روش را میتوانید مشاهده کنید:

$$C(r) = \int_{t_n}^{t_f} T(t)\, \sigma\big(r(t)\big)\, c(r(t).d)dt.\ where T(t) = exp(-\int_{t_n}^{t} \sigma\big(r(s)\big)ds)$$

در این رابطه، $r(t) = \mathbf{o} + t\mathbf{d}$، یک پرتو از سمت دوربین به نقطه‌ای از صحنه است، $T(t)$ تابعی است که مشخص میکند از ابتدای پرتو تا نقطه‌ای بر روی پرتو r که به صورت $r(t)$ محاسبه می‌شود، چه میزان چگالی دیده ایم. این تابع مشخص می کند که آیا نقطه در حال محاسبه، از سمت دوربین قابل مشاهده است یا خیر. در نهایت، $C(r)$ رنگ نقطه پیکسلی را مشخص می کند که پرتوی r از دوربین به سمت صحنه و از مسیر آن پیکسل تابش شده است.برای درک بهتر این جمله می‌توانید به بخش b در شکل ۲ توجه کنید.

همانطور که مشخص است، تولید یک تصویر کامل نیازمند ارسال یک پرتو به ازای هر پیکسل تصویر خروجی است. به ازای هر پرتو، انتگرال داده شده محاسبه می‌شود و رنگ متناسب با آن پیکسل به‌دست می‌آید. محاسبه این انتگرال نیازمند نوعی تخمین است که در این مقاله از روش Quadrature استفاده می‌شود. مشکلی که در این بخش وجود دارد این است که اگر برای تخمین این انتگرال، t هایی با فاصله یکسان را انتخاب کنیم و مقدار تابع را در این نقاط محاسبه کنیم، عملا تابع پیوسته ای که داشتیم را گسسته کردیم و فقط در نقاط مشخصی از آن استفاده می کنیم. راه حلی که نویسنده های مقاله برای رفع این مشکل ارائه داده اند به این صورت است که به جای نمونه برداری یکسان، از نمونه برداری گروهی[11] استفاده میکنند. در این نوع نمونه‌برداری، ابتدا بازه $[t_n, t_f]$ را به N بازه با اندازه یکسان تقسیم میکنیم، سپس، به صورت تصادفی از هر بازه یک نقطه انتخاب می کنیم. با این کار از دید کلی، نمونه برداری یکسانی داریم، اما از دید نزدیک‌تر، تصادفی بودن را به نمونه برداری وارد کرده ایم که باعث می‌شود تمام اطلاعات موجود در F_θ مورد استفاده قرار بگیرند.

این نمونه برداری همچنان قابل بهبود است. نمونه برداری فعلی به تمام صحنه وزن یکسان نسبت می‌دهد و نقاط را به صورت یکنواخت پراکنده می کند. اما بهتر است نقاطی از صحنه که فضای خالی هستند تعداد نمونه‌های کمتری داشته باشند و نقاط مهم صحنه مانند موضوع اصلی صحنه(مثلا ماشین اسباب بازی در شکل ۲) تعداد نمونه‌های بیشتری داشته باشند. برای حل این مشکل، نویسنده‌های این مقاله روشی ارائه دادند که طبق آن به طور همزمان دو نسخه از مدل را آموزش می‌دهیم، یک نسخه با جزئیات کم[12] و یک نسخه با جزئیات زیاد[13]. سپس در مرحله اول با کمک روش نمونه برداری گروهی تعداد N_c نمونه را از روی مدل با جزئیات کم برمی‌داریم. در مرحله دوم، تعداد N_f نمونه

[11] Stratified Sampling

[12] Coarse

[13] Fine

را از روی مدل با جزئیات بالا با کمک وزن‌های بدست آمده از مدل با جزئیات کم برمی‌داریم. تصویر نهایی در این حالت از ترکیب نمونه های N_f و N_c در کنار هم بدست می‌آید. در این حالت نمونه های مربوط به N_c به طور یکنواخت در طول پرتو پخش شده اند و نمونه های مربوط به N_f با تمرکز بیشتری در نقاطی که موضوعات صحنه قرار دارند، پخش شده اند.

	Input	#Im.	L	(N_c , N_f)	PSNR↑	SSIM↑	LPIPS↓
1) No PE, VD, H	xyz	100	-	(256, -)	26.67	0.906	0.136
2) No Pos. Encoding	$xyz\theta\phi$	100	-	(64, 128)	28.77	0.924	0.108
3) No View Dependence	xyz	100	10	(64, 128)	27.66	0.925	0.117
4) No Hierarchical	$xyz\theta\phi$	100	10	(256, -)	30.06	0.938	0.109
5) Far Fewer Images	$xyz\theta\phi$	25	10	(64, 128)	27.78	0.925	0.107
6) Fewer Images	$xyz\theta\phi$	50	10	(64, 128)	29.79	0.940	0.096
7) Fewer Frequencies	$xyz\theta\phi$	100	5	(64, 128)	30.59	0.944	0.088
8) More Frequencies	$xyz\theta\phi$	100	15	(64, 128)	30.81	0.946	0.096
9) Complete Model	$xyz\theta\phi$	100	10	(64, 128)	**31.01**	**0.947**	**0.081**

جدول ۱ ــ نتایج آزمایش‌های مختلف مدل NeRF برروی مجموعه داده تصاویر مصنوعی

در جدول ۱ می‌توانید نتایج مدل را به ازای مقادیر مختلف N_f و N_c مشاهده کنید. مدل بر روی مجموعه داده تصاویر مصنوعی که توسط نویسنده‌های مقاله تهیه شده اجرا شده است و نتیجه ای که مشاهده می کنید میانگین امتیازهای مدل به ازای هر ۸ صحنه موجود در مجموعه داده است.

۴ـ۲. آموزش شبکه

دانستن دو بخش قبل برای متوجه شدن نحوه آموزش مدل ضروری است. به همین دلیل، به این ترتیب به توضیح بخش ها پرداختیم.

قبل از شروع یادگیری بهتر است کمی با مجموعه داده های این مسئله آشنا شویم. همان‌طور که تا الان باید بدانید، هدف این شبکه، تولید تصاویری از زوایای تابه‌حال دیده نشده از یک صحنه، با داشتن تعداد محدودی تصویر از آن صحنه است. تعداد تصویر مورد نیاز برای آموزش شبکه، به میزان پیچیدگی صحنه بستگی دارد اما معمولا بین ۲۰ الی ۱۰۰ عدد است. این تصاویر از یک صحنه مشترک اما از زوایای مختلف گرفته شده اند(به بخش a شکل ۱ رجوع کنید). نکته مهم دیگری که در مورد داده‌های ورودی این روش وجود دارد این است که این روش، به عنوان ورودی علاوه بر خود تصاویر، به زاویه دوربینی که این تصاویر را گرفته نیز نیاز دارد. دلیل این امر پس از بیان روش یادگیری شبکه مشخص می‌شود. در حال حاضر به این نکته بسنده می‌کنیم که زوایای دوربین به کمک روش‌های پردازش تصویر آماده مانند ساختار از حرکت[۱۴] یا SfM[۷] محاسبه می‌شود. علاوه بر این تعدادی از تصاویر مورد استفاده در این مقاله، به صورت مصنوعی و از طریق نرم‌افزارهای سه‌بعدی تولید شده اند که برای این تصاویر به دلیل دسترسی دقیق به موقعیت دوربین، به طور مستقیم از موقعیت دوربین‌ها استفاده می‌شود.

برای آموزش این مدل، هر تصویر به عنوان ورودی به همراه زاویه دوربین متناسب، به شبکه داده می‌شود. شبکه طبق توضیحات داده شده در بخش ۳ـ۲ با استفاده از وزن‌های فعلی شبکه، به تولید دو تصویر با جزئیات پایین و جزئیات بالا می‌پردازد. در ادامه طبق رابطه زیر، خطای متناظر تولید می‌شود.

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

محاسبه این خطا با مقایسه کردن تصویر تولید شده توسط شبکه و تصاویر ورودی شکل می‌گیرد. با استفاده از الگوریتم بازنشر خطا، وزن‌های شبکه در جهت کاهش این خطا تنظیم می‌شوند. انتظار می‌رود پس از گذشت حدود ۱۰۰هزار الی ۳۰۰ هزار مرحله، تصاویر قابل قبولی تولید شوند.

نویسندگان این مقاله با استفاده از یک کارت گرافیک Nvidia v100 به آموزش این شبکه پرداخته اند و زمان اتمام یادگیری برای این مدل را حدود ۱ الی ۲ روز گزارش کرده‌اند.حجم وزن‌های این مدل به ازای هر صحنه حدود 5MB گزارش شده که حتی در مقایسه با حجم تصاویر ورودی نیز حجم بسیار کمتری است، فارغ از اینکه این وزن‌ها قادر به تولید تصاویر بیشتری نیز هستند. زمان اجرای این شبکه و تولید یک تصویر از نمای جدید، پس از آموزش، بر اساس اندازه تصویر، بین ۱ الی ۳۰ ثانیه طول می‌کشد که نشان می‌دهد این روش نمی‌تواند به صورت بی‌درنگ[۱۵]استفاده شود.

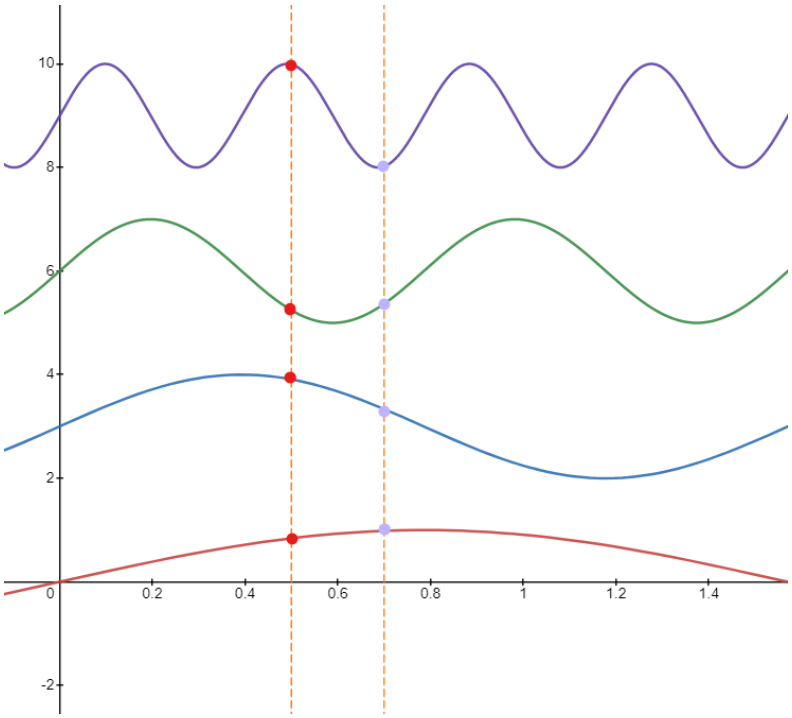
[14] Structure from motion

[15] Realtime

۵ـ۲. کدگذاری مکانی[16]

می‌دانیم یکی از کاربردهای اصلی شبکه های عصبی، تخمین توابع است. با این وجود، تحقیقات اخیر نشان داده است که شبکه‌های عصبی تمایل بیشتری به تخمین زدن یک تابع به صورت نَرم و فرکانس پایین دارند[۸]. آزمایش-های عملی نویسنده‌های این مقاله نیز این فرضیه را تقویت می‌کنند. نویسنده‌ها متوجه شدند که صرفا با دادن داده‌ها به صورت $xyz\theta\phi$، شبکه توانایی بازنمایی ویژگی‌های فرکانس بالا را ندارد. در بخش d شکل ۴ می‌توانید این اتفاق را مشاهده کنید.

راه حلی که برای این مشکل ارائه شده است کدگذاری مکانی نام دارد. این نوع کدگذاری به این شکل عمل میکند که تابع F را با یک تابع γ ترکیب میکنیم. حاصل این ترکیب، انتقال مولفه‌های F به بعد بالاتری است. در این حالت مدل نه تنها یک ورودی را در اختیار دارد، بلکه نسخه های مختلف آن ورودی با فرکانس‌های مختلف را در اختیار دارد. مدل می‌تواند انتخاب کند در هر لحظه از کدام نسخه آن ورودی استفاده کند. نسخه‌های فرکانس بالا، با تغییر اندکی در ورودی، تغییر زیادی در خروجی را نمایش می‌دهند و نسخه‌های فرکانس پایین، با تغییر زیاد در ورودی، تغییر اندکی در خروجی را نمایش می‌دهند. در شکل ۵ می‌توانید این اثر را مشاهده کنید.



شکل ۵ ـ نمایشی از کدگذاری مکانی

[16] Positional Encoding

معماری مبدل‌ها[17] [۶] نیز از یک روش مشابه اما برای هدفی متفاوت استفاده کرده است. در معماری مبدل‌ها، به دلیل عدم وابستگی داده ورودی به موقعیت مکانی کلمات در جمله، نیاز است به نحوی موقعیت مکانی کلمات وارد شبکه شود. آنجا این کار را با استفاده از کدگذاری مکانی به شکل مشابه انجام می‌دهند. اما در این کار، دلیل انجام کدگذاری مکانی، انتقال داده ورودی به بعد بالاتری است تا شبکه بتواند جزئیات فرکانس‌بالای صحنه را بهتر دریافت و ذخیره کند.

[17] Transformers

۶ـ۲. نتایج و جمع‌بندی

در جدول ۲ می‌توانید عملکرد این روش را در مقایسه با سایر روش‌های مطرح، با استفاده از مجموعه داده‌های "Diffuse Synthetic 360", "Realistic Synthetic 360" و "Real Forward-Facing" مشاهده کنید.

Method	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

جدول ۲ — نتایج روش NeRF در مقایسه با سایر روش‌ها

مشاهده می‌شود که این روش نسبت به اکثر روش‌های دیگر در همه مجموعه داده‌ها برتری دارد. به گفته نویسنده‌های مقاله، با اینکه در معیار LPIPS که مدل LLFF امتیاز بهتری گرفته، با دیدن ویدیوهای مربوط به این دو روش مشاهده می‌شود که روش NeRF تصاویر یکسان تری را برای فریم‌های متوالی تولید می‌کند و از دید انسان‌ها روش بهتری است. در شکل ۵ می‌توانید تعدادی از تصاویر تولید شده توسط روش NeRF و سایر روش‌ها را برای مقایسه مشاهده کنید.

شکل ٦ ـ نمونه تصاویر تولید شده توسط روش NeRF و سایر روش‌ها

۳. بهبود NeRF

این شبکه، با وجود تمامی ویژگی‌های نابی که دارد، از تعدادی مشکل جدی رنج می‌برد. برخی از این مشکلات، کلی هستند و در این بخش اشاره ای به آنها خواهیم داشت. تعدادی مشکل دیگر نیز وجود دارند که در کاربردهای خاص بوجود می‌آیند. مثال هر دو نوع مشکل را در بخش‌های بعد گزارش خواهیم دید. در طی دو سال اخیر، بیش از ۲۰ مقاله برای بهبود NeRF نوشته شده است. این مقالات عمدتاً در زمینه رفع همین مشکلات راه‌های را پیشنهاد داده اند.

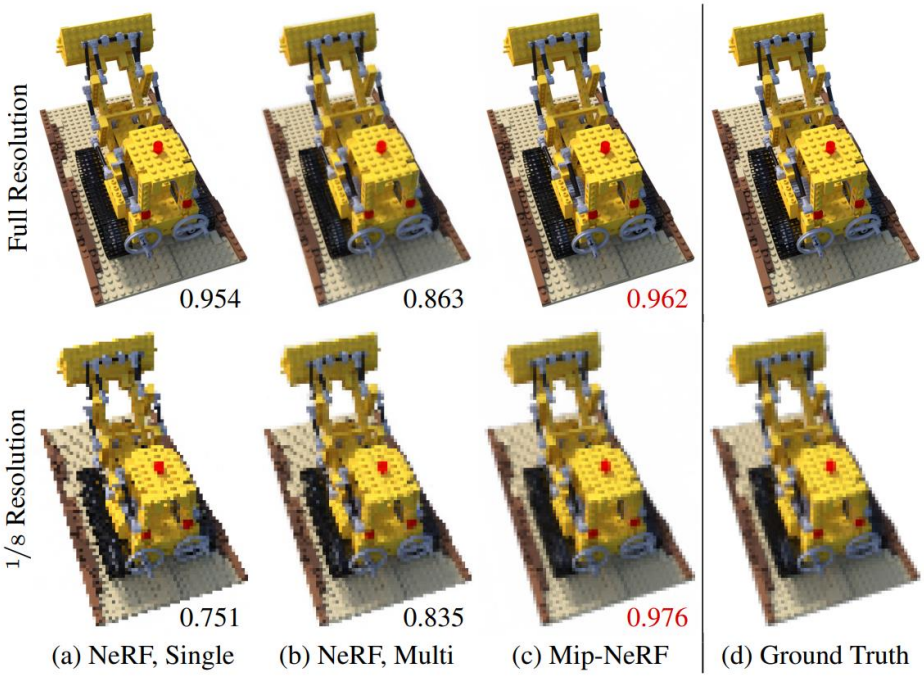
مشکل‌هایی که در این گزارش به آنها خواهیم پرداخت:

۱. تولید تصاویر در مقیاس‌های مختلف (*شبکه NeRF معمولی محدود به تولید تصاویر در اندازه اصلی است*)
۲. بازنمایی صحنه‌های بسیار وسیع (*در حد چندین خیابان به هم متصل*)
۳. نداشتن تعداد کافی داده آموزشی برای یک صحنه (*به عنوان مثال استفاده از فقط ۲ نما به عنوان داده آموزش*)
۴. تسریع مرحله اجرای NeRF (قابلیت اجرای بی‌درنگ)

۱ـ۳. تولید تصاویر در مقیاس‌های مختلف (Mip-NeRF)

۱ـ۱ـ۳. بیان مشکل و مقدمه

از جمله مشکلات اصلی روش NeRF، محدود بودن به تولید تصاویر در اندازه مشخص و از پیش تعیین شده است. به طورکلی تصاویر تولیدی توسط NeRF با وجود اینکه قابلیت تولید شدن از زوایای مختلف را دارند، اما قابلیت تولید شدن از فاصله‌های مختلف را ندارند. این امر باعث می‌شود حرکت کردن سه بعدی در یک محیط امکان پذیر نباشد و بسیاری از کاربردهای بلقوه NeRF عملی نباشند.



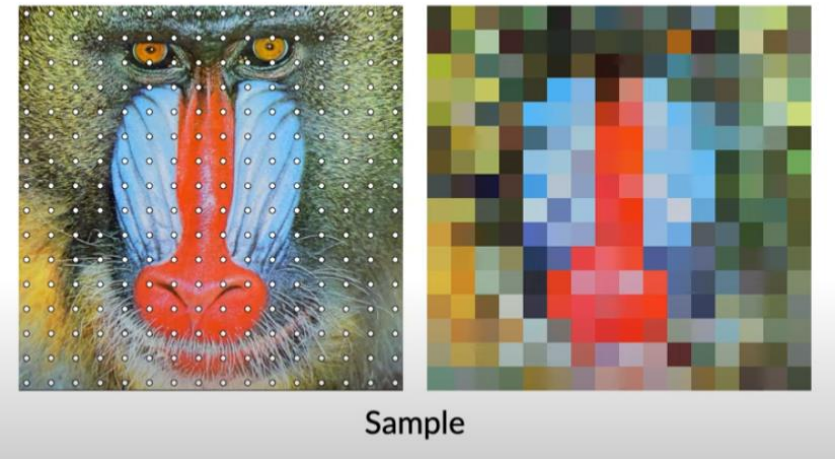
شکل ۷ـ مقایسه NeRF و Mipp-NeRF(معیار SSIM)

اگر به بخش a شکل ۷ نگاه کنید متوجه خواهید شد که NeRF، برخلاف خروجی بسیار زیبایی که در اندازه اصلی می‌دهد، در اندازه های بزرگ‌تر یا کوچکتر عملکرد ضعیفی دارد و نتیجه اش دچار دندانه‌دارشدن[۱۸] شدیدی می‌شود. یکی از تکنیک‌های بسیار ساده برای بهبود دندانه‌دارشدن نتایج NeRF، آموزش دادن آن توسط تصاویری با اندازه‌های مختلف است. با این کار شبکه نهایی می‌تواند با درون‌یابی بین تصاویر با اندازه‌های مختلف، تصاویرجدید را تولید کند. این روش، تا حدی مشکل دندانه‌دارشدن در مقیاس های مختلف را رفع می‌کند اما تصاویر در مقیاس اصلی را نیز تار می‌کند.

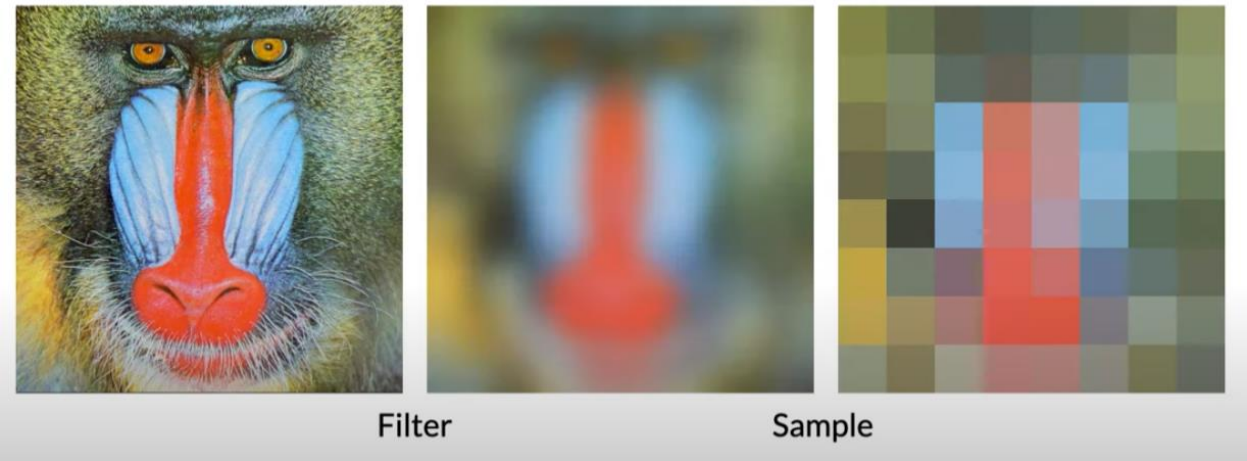
[18] Aliasing

معماری Mip-NeRF[۲] با الهام گیری از روش Mip-map که در پردازش تصویر برای تغییر اندازه تصاویر بدون ایجاد دندانه‌دارشدن در تصاویر دوبعدی است توانسته این محدودیت NeRF را بردارد و بتواند با دقت بالایی تصاویر در ابعاد مختلف را با روشی مشابه NeRF تولید کند.

اگر یک تصویر را به ساده‌ترین شکل ممکن تغییراندازه دهیم با نتیجه نامطلوبی مانند نتیجه شکل ۸ مواجه خواهیم شد. یک راه حل برای این مشکل این است که قبل از تغییراندازه دادن تصویر، آنرا به کمک یک فیلتر پایین‌گذر مانند فیلتر گاوسی، فیلتر کنیم. سپس تصویر را تغییر اندازه دهیم. در شکل ۹ میبینیم که این روش تاثیر مثبتی بر روی نتیجه می‌گذارد. به این تکنیک پیش‌فیلترکردن[۱۹] گفته می‌شود.



شکل ۸ ــ نمونه‌برداری تنها



شکل ۹ ــ نمونه‌برداری پس از اعمال فیلتر پایین‌گذر

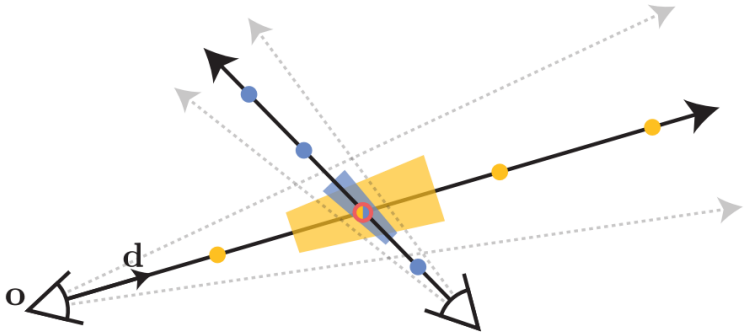
[19] Prefiltering

دلیل این‌که فیلترکردن تصویر به کمک یک فیلتر پایین‌گذر دندانه‌دارشدن را رفع می‌کند این است که فیلترپایین گذر، بالاترین فرکانس موجود در تصویر را به فرکانس پایین‌تری نسبت به تصویر اصلی تبدیل می‌کند. این اتفاق باعث می‌شود شرط Nyquist برقرار شود و دندانه‌دارشدن رخ ندهد.

روش پیش‌فیلترکردن، یک روش بسیار سنگین است. زیرا هر بار قبل از تغییر اندازه نیاز است تصویر یک‌بار فیلتر شود. برای حل این مشکل روش Mip-map معرفی شد. این روش با اعمال فیلترهایی با شدت‌های متفاوت به تصویر، یک سلسه‌مراتب از تصویر را از قبل تولید میکند. در هنگام نیاز، این تصاویر از قبل محاسبه شده، با کمک درون‌یابی tri-linear یا سه خطی، در سه جهت افقی، عمودی و در راستای مقیاس با هم ترکیب می‌شوند و یک تصویر عاری از دندانه‌دارشدن را در مقیاس دلخواه تولید می‌کند.

۲ـ۱ـ۳. راه حل مشکل

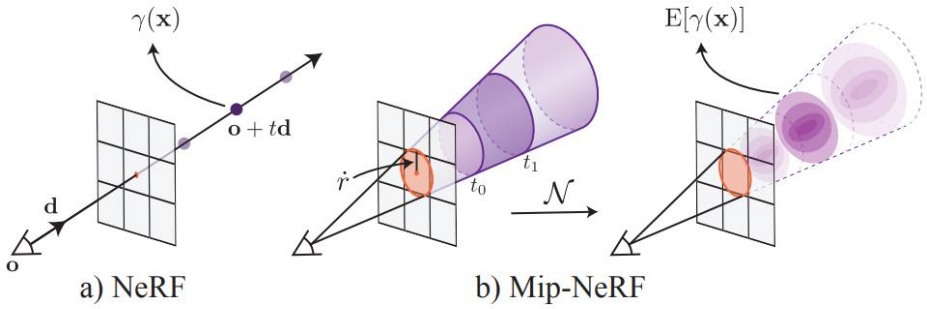
روش Mip-NeRF با الهام از این روش، سعی می‌کند مدل NeRF را غیروابسته به مقیاس کند و بتواند خروجی موردنظر را در هر مقیاسی بدون دندانه‌دارشدن تولید کند. البته پیاده سازی Mip-NeRF با Mip-Map بسیار متفاوت است زیرا به علت در دست نداشتن صحنه نهایی، نمی‌توان مانند روش Mip-map نسخه‌های متعدد و در مقیاس مختلفی از صحنه را تولید کرد. در عوض، شبکه Mip-NeRF در حین آموزش، یک نسخه پیش‌فیلترشده از صحنه را می‌آموزد و در هر لحظه میتوان آنرا در یک مقیاس دلخواه اجرا کرد.



شکل ۱۰ ـ مشکل نمونه‌برداری در روش NeRF

در شکل ۱۰ می‌توانیم منشاء مشکل دندانه‌دارشدن NeRF را مشاهده کنیم.چون در NeRF نمونه برداری را در راستای یک پرتوی بی‌نهایت باریک انجام می‌دهیم، نمونه‌ها هیچ ایده ای راجع به شکل و اندازه حجمی که با پرتو برخورد کرده ندارند. برای اثبات این امر دو پرتوی مشخص شده در شکل ۱۰ را مشاهده کنید. این دو پرتو هر دو در یک نقطه مشترک با حجم برخورد کرده اند و از دید این دو پرتو، این نقطه از حجم کاملا یکسان است. بر خلاف NeRF، در Mip-NeRF به جای این پرتوهای بینهایت باریک، از هر پیکسل یک مخروط به سمت صحنه ارسال می‌شود و برای نمونه برداری از صحنه، به جای انتخاب کردن تعدادی نقطه، تعدادی قطاع مخروط انتخاب می‌شوند.در شکل ۱۰ مشخص شده که این

قطاع‌ها(که در تصویر با استفاده از ذوزنقه‌های زرد و آبی نشان داده شده اند)، در یک نقطه از حجم، به شکل و اندازه حجم نیز توجه می‌کنند.



شکل ۱۱ ـ مقایسه نحوه کارکرد روش‌های NeRF و Mip-NeRF

همان‌طور که بیان شد و در شکل ۱۱ نیز مشاهده می‌شود، روش Mip-NeRF از تعدادی قطاع‌مخروط برای نمونه برداری استفاده می‌کند.نویسنده‌های این مقاله، این قطاع‌ها را با استفاده از یک تابع گاوسی جایگزین کرده‌اند تا محاسبات ساده تر شود و برای تعدادی از روابط مورد نیاز فرم بسته به‌وجود آید. همچنین نویسنده‌های این مقاله کدگذاری مکانی موجود در روش NeRF را با نسخه ای متناسب با نمونه‌های گاوسی، به اسم کدگذاری مکانی انتگرالی[20] یا IPE جایگزین کرده اند.

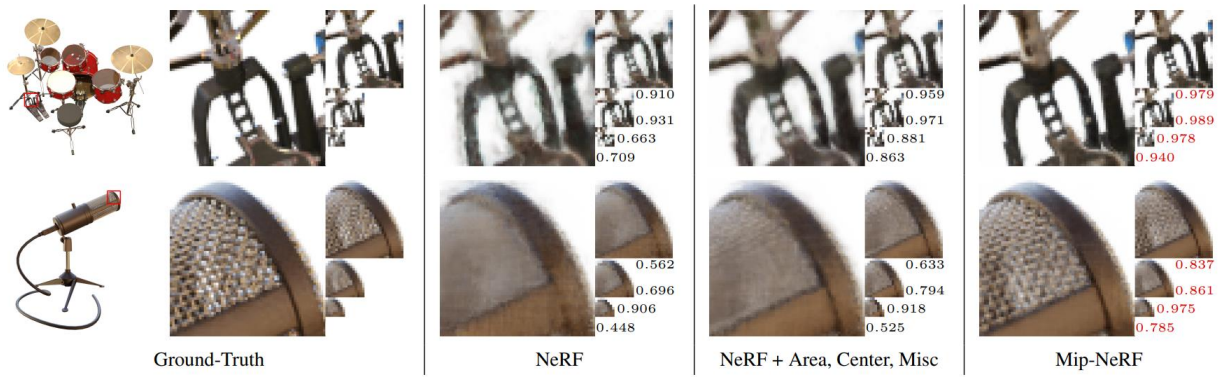
همچنین، به دلیل این‌که Mip-NeRF، صحنه را در مقیاس‌های مختلف می‌آموزد، نیازی به آموختن دو نسخه متفاوت با جزئیات بالا و با جزئیات پایین از مدل نیست. اگر به خاطر داشته باشید این دو مدل را به این دلیل آموختیم که بتوانیم نمونه برداری نقاط را هوشمندانه تر کنیم. در Mip-NeRF خبری از آن نقاط نیست پس نیازی به یادگیری دو مدل مجزا هم وجود ندارد. به دلیل این ساده سازی، روش Mip-NeRF به میزان ۷ درصد سریع‌تر از NeRF است و به میزان ۵۰ درصد، پارامترهای کمتری دارد.

تمام این ویژگی ها، معماری Mip-NeRF را به جایگزین بسیار مناسبی برای NeRF در اکثر کاربردها تبدیل کرده است. در ادامه این گزارش خواهیم دید که حتی معماری‌های دیگری که در تلاش برای بهبود بخشی از NeRF بوده‌اند نیز، به عنوان مدل پایه از Mip-NeRF بجای NeRF استفاده کرده اند.

[20] Integerated Positional Encoding

۳ـ۱ـ۳. نتایج

در شکل ۱۲ می‌توانید تفاوت روش NeRF و Mip-NeRF را به ازای تغییراندازه‌های مختلف، مشاهده کنید. واضح است که روش Mip-NeRF در این زمینه عملکرد بسیار بهتری دارد. اعداد موجود در تصویر معیار SSIM متناظر با هر مقیاس هستند که هر چه به عدد ۱ نزدیک‌تر باشد یعنی تصویر تولید شده به تصویر مرجع شباهت ساختاری بیشتری دارد.



شکل ۱۲ ـ مقایسه عملکرد Mip-NeRF و NeRF بر روی دو صحنه مصنوعی

در جدول ۳ می‌توان به صورت دقیق‌تر، عملکرد Mip-NeRF و بهبود آن نسبت به NeRF را مشاهده کرد.

	PSNR ↑				SSIM ↑				LPIPS ↓						
	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res.	Full Res.	1/2 Res.	1/4 Res.	1/8 Res	Avg. ↓	Time (hours)	# Params
NeRF (Jax Impl.) [11, 30]	31.196	30.647	26.252	22.533	0.9498	0.9560	0.9299	0.8709	0.0546	0.0342	0.0428	0.0750	0.0288	3.05 ± 0.04	1,191K
NeRF + Area Loss	27.224	29.578	29.445	25.039	0.9113	0.9394	0.9524	0.9176	0.1041	0.0677	0.0406	0.0469	0.0305	3.03 ± 0.03	1,191K
NeRF + Area, Centered Pixels	29.893	32.118	33.399	29.463	0.9376	0.9590	0.9728	0.9620	0.0747	0.0405	0.0245	0.0398	0.0191	3.02 ± 0.05	1,191K
NeRF + Area, Center, Misc.	29.900	32.127	33.404	29.470	0.9378	0.9592	0.9730	0.9622	0.0743	0.0402	0.0243	0.0394	0.0190	2.94 ± 0.02	1,191K
Mip-NeRF	32.629	34.336	35.471	35.602	0.9579	0.9703	0.9786	0.9833	0.0469	0.0260	0.0168	0.0120	0.0114	2.84 ± 0.01	612K
Mip-NeRF w/o Misc.	32.610	34.333	35.497	35.638	0.9577	0.9703	0.9787	0.9834	0.0470	0.0259	0.0167	0.0120	0.0114	2.82 ± 0.03	612K
Mip-NeRF w/o Single MLP	32.401	34.131	35.462	35.967	0.9566	0.9693	0.9780	0.9834	0.0479	0.0268	0.0169	0.0116	0.0115	3.40 ± 0.01	1,191K
Mip-NeRF w/o Area Loss	33.059	34.280	33.866	30.714	0.9605	0.9704	0.9747	0.9679	0.0427	0.0256	0.0213	0.0308	0.0139	2.82 ± 0.01	612K
Mip-NeRF w/o IPE	29.876	32.160	33.679	29.647	0.9384	0.9602	0.9742	0.9633	0.0742	0.0393	0.0226	0.0378	0.0186	2.79 ± 0.01	612K

جدول ۳ ـ عملکرد Mip-NeRF در مقایسه با NeRF به ازای میزان مختلف تغییر اندازه

۲ـ۳. بازنمایی صحنه‌های وسیع (Block-NeRF)

۱ـ۲ـ۳. بیان مشکل و مقدمه

از جمله مشکلات دیگر شبکه NeRF، بازنمایی صحنه‌های بسیار وسیع است. همانطور که در تصاویر موجود در مقاله NeRF مشاهده میکنید، اکثر صحنه‌های مورد بررسی قرار داده شده، حدودا چند متر عرض دارند. محققان خیلی زود متوجه شدند که بازنمایی صحنه های بزرگ‌تر مانند یک اتاق، یک خانه، یا در موارد خاص یک محله، با استفاده از NeRF بسیار نامطلوب یا غیرممکن است.

پژوهشگران کمپانی **Google** و کمپانی تولید کننده اتومبیل‌های خودران **Waymo** در یک همکاری، تصمیم گرفتند با بهبود شبکه NeRF آن‌را برای بازنمایی صحنه‌های بسیار وسیع مانند یک محله(در این مثال حدود ۱۰۰۰ متر در ۶۰۰ متر) به کارگیرند. شبکه ای که توسط آنها ارائه شد،میدان‌های تابش عصبی بلوکی[3] یا Block-Nerf نام دارد.

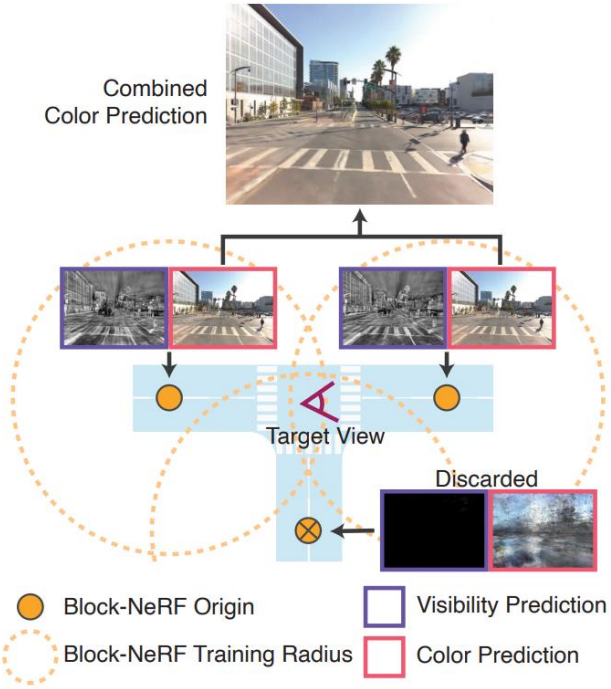
این مقاله یک مقاله بسیار کامل است. نویسنده‌های این مقاله، داده‌های مربوط این مقاله را نیز جمع آوری کرده اند که طبق گفته خودشان، حدود ۷ ماه به طول انجامیده و حدود ۳ میلیون تصویر از نواحی مختلف یک محله در سان‌فرانسیسکو فراهم شده. در این مقاله تعداد زیادی ایده جدید به شبکه NeRF افزوده شده که باعث می‌شود بسیاری از مشکلات در زمینه بازنمایی صحنه‌های سخت را رفع کند.

راه بسیار ساده‌ای که به ذهن می‌رسد این است که پیچیدگی مدل را افزایش دهیم تا مدل بتواند صحنه بزرگتری با جزئیات بیشتری را بازنمایی کند. اما این روش تعداد بسیار زیادی مشکل دارد. مشکل اول این است که ممکن است مدل به اندازه‌ای بزرگ شود که قابل اجراشدن برروی یک رایانه نباشد. مشکل دوم زمان اجرا است، وقتی صحنه خیلی بزرگی را در وزن‌های مدل ذخیره کرده باشیم، نیاز است تعداد نمونه‌های بسیار زیادی برداریم تا بتوانیم تمام صحنه را پوشش دهیم که این باعث کند شدن بیش از حد زمان اجرا و آموزش می‌شود.مشکل دیگر مدیریت کردن این مدل است. اگر در حین کار تصمیم گرفته شود که میخواهیم صحنه را به اندازه یک خیابان بزرگ تر کنیم، نیاز است کل مدل از ابتدا آموزش بی‌یابد. یک مشکل خیلی مهم دیگر مخصوصا برای صحنه‌های بسیار بزرگ این است که به احتمال بسیار زیاد، تصاویر موجود از آن صحنه، در زمان‌های مختلف و در شرایط نوری متفاوتی تهیه شده اند. این مشکل برای مسئله تعریف شده در مقاله ذکر شده به شکل جدی ای وجود دارد زیرا تصاویر محیط، در طی ۷ ماه و در ساعات مختلف شبانه روز جمع آوری شده اند. این امر باعث می‌شود تصاویر مکان‌های نزدیک به هم، از لحاظ شرایط نوری یا حتی شرایط ساختاری تفاوت بسیاری داشته باشند. در مورد صحنه های شهری، وجود عابر پیاده و اتومبیل ها نیز یک مشکل بزرگ دیگر است.

در مقاله Block-NeRF، تمام مشکلات ذکر شده به همراه تعدادی دیگر از مشکلات به زیبایی حل شده اند و نتیجه خارق العاده است! متاسفانه شرح تک تک این راه‌حل ها در این گزارش نمی‌گنجد و به بیان تعدادی از مهم‌ترین ایده‌های این مقاله بسنده می‌کنیم.

۲ـ۲ـ۳. تقسیم شبکه بزرگ به تعدادی شبکه کوچک‌تر

ایده اصلی این روش که اسم آن هم از همان‌جا می‌آید، تقسیم بندی شبکه NeRF بسیار بزرگ، به تعدادی شبکه NeRF کوچک تر است. در این مقاله، هر کدام از این شبکه‌های کوچک‌تر، وظیفه بازنمایی یک تقاطع را در یک محله دارند. با آموزش مستقل این شبکه‌های کوچک‌تر و در کنارهم گذاشتن آنها به هنگام نیاز، تعداد زیادی از مشکلات بیان شده رفع می‌شوند. اولین اثر این روش این است که آموزش شبکه‌های کوچکتر کار بسیار ساده‌تری است. دومین اثر این است که تولید نماهای جدید بسیار سریعتر انجام می‌شود زیرا برای تولید هر نما، به تعداد محدودی از این شبکه‌های کوچک نیاز است. به شکل ۱۳ توجه کنید. همچنین در آینده، می‌توان این شبکه را به آسانی گسترش داد. کافی‌است از سایر نقاط محله یا حتی محله های همسایه، تعداد کافی تصویر گرفته شود، شبکه‌های کوچک متناظر، ساخته شوند و آموزش بینند، سپس به مجموعه شبکه‌های اصلی افزوده شوند.



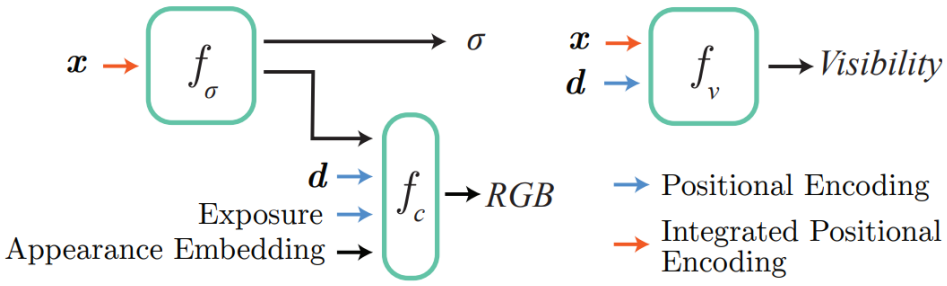
شکل ۱۳ ـ تولید یک نمای جدید به کمک Block-NeRF

۳ـ۲ـ۳. انتخاب بلاک‌ها

در شکل ۱۳ شعاع تحت پوشش هر شبکه‌کوچک با خطچین نارنجی نمایش داده شده است.این شبکه‌های کوچک حدود ۵۰ درصد روی‌هم‌رفتگی دارند که باعث می‌شود تولید تصاویر در مرزهای بین شبکه ها بهبود یابد. به هنگام تولید نماهای جدید، مدل می‌تواند بر اساس موقعیت دوربین، فقط شبکه‌های کوچک مورد نیاز را فراخوانی کند و از سایر شبکه ها صرف نظر کند که این اتفاق باعث افزایش سرعت تولید نماهای جدید می‌شود. تصاویر سطح‌خاکستری موجود در شکل ۱۳ یکی دیگر از ابداع های این مقاله هستند. این تصاویر، تصاویر میدان دید هستند. این تصاویر مشخص می‌کنند که در یک موقعیت خاص و با یک زاویه دید خاص، کدام شبکه‌ها قابل رویت هستند و کدام شبکه‌ها کاملا از دید خارج هستند. با استفاده از این تصاویر، می‌توان شبکه‌های بی استفاده را به هنگام تولید نماهای جدید در نظر نگرفت حتی اگر این شبکه‌ها از لحاظ موقعیت جغرافیایی در محدوده دوربین باشند و با استفاده از تکنیک قبل حذف نشده باشند و به این ترتیب سرعت تولید نماهای جدید باز هم افزایش می‌یابد.

٤ـ٢ـ٣. معماری شبکه و حل تعدادی دیگر از مشکلات

معماری شبکه‌های Block-NeRF، بر اساس شبکه Mip-NeRF بنا شده است. استفاده از این شبکه‌ی بهبود یافته، اجازه حرکت در محیط های تولید شده را می‌دهد و در عین‌حال پیچیدگی مدل را تا حد زیادی کم می‌کند. این امر برای همچین مسئله‌ای که در آن تعداد بسیار زیادی از این شبکه‌ها موجود هستند بسیار مفید است. مانند سایر شبکه‌های بر پایه NeRF، این شبکه نیز خروجی σ را صرفاً بر اساس ورودی x تولید می‌کند و خروجی $c = (r.g.b)$ را بر اساس همه ورودی‌ها تولید می‌کند. این کار خاصیت استقلال از نما را برای ساختار اجسام و وابستگی به نما را برای رنگ ایجاب می‌کند.



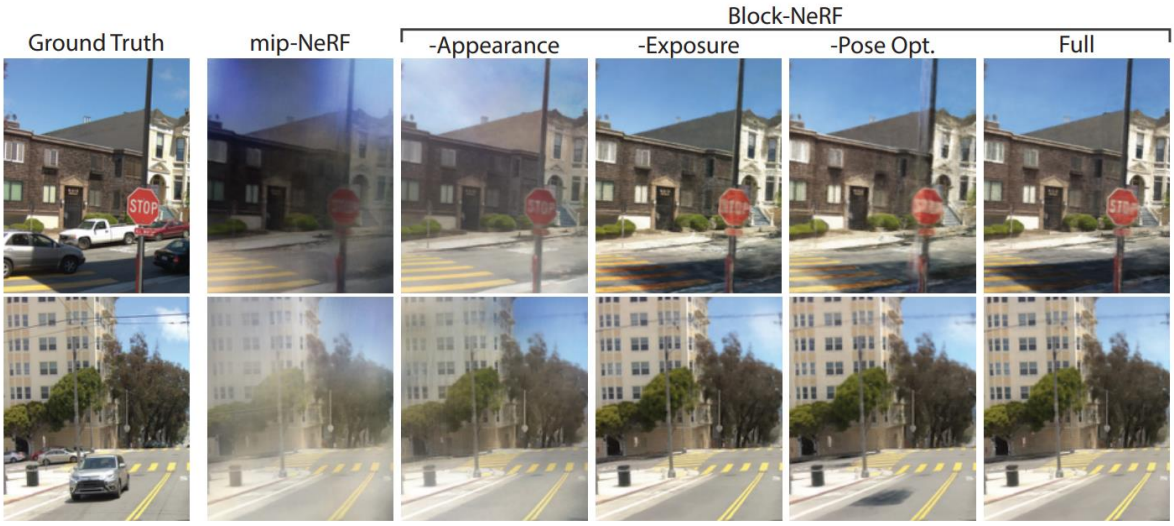
شکل ١٤ ـ معماری شبکه‌های Block-NeRF

همانطور که حتما متوجه شدید، این شبکه برای تخمین خروجی رنگ، علاوه بر ورودی های رایج، یک سری ورودی دیگر نیز دریافت می‌کند. توضیح کاربرد و نحوه عملکرد هر کدام از این ورودی‌ها بسیار زمان‌بر خواهد بود اما در بخش نتایج می‌توانید اثر بودن یا نبودن آنها را مشاهده کنید.

در کنار شبکه‌های تولید خروجی‌های مورد نیاز NeRF، در شکل ۱۴ یک شبکه کوچک دیگر نیز داریم. این شبکه که به طور همزمان با شبکه تولید σ آموزش داده می‌شود، وظیفه تعیین قابل رویت بودن یک نقطه را در شبکه فعلی بر عهده دارد. این شبکه با دریافت $x.d$، یک تصویر سطح‌خاکستری مانند تصاویر موجود در شکل ۱۳ تولید می‌کند که میزان رویت‌پذیر بودن نقاط مختلف صحنه را از مکان و زاویه دید فعلی مشخص می‌کنند. اگر میزان رویت‌پذیری میانگین در یک بلوک، از یک حد آستانه کمتر باشد، آن بلاک در پروسه تولید نمای جدید شرکت داده نخواهد شد.

۵ـ۲ـ۳. نتایج

ابتدا به شکل ۱۵ توجه کنید. ستون دوم از سمت چپ، نتیجه تولید یک نمای جدید، با استفاده از شبکه Mip-NeRF خام است. همانطورکه در بخش‌های قبل اشاره شد، به دلیل تفاوت میزان روشنایی و وضعیت تصاویر مختلف، لازم است با استفاده از تکنیک‌هایی، این اثرات را خنثی کرد. اگر به ستون آخر از سمت چپ نگاه کنیم، نتیجه نهایی شبکه Block-NeRF با تمام امکاناتش را مشاهده میکنیم. ستون پنجم از سمت چپ، نتیجه نهایی شبکه در حالتی که مرحله Pose Optimization را نادیده گرفته است. با وجود کالیبره بودن دوربین‌ها و تشخیص دقیق موقعیت آنها نسبت به یکدیگر و محیط، باز هم مقداری خطا در مکان دقیق اجسام وجود دارد. نویسندگان این مقاله برای رفع این مشکل، در حین آموزش شبکه، یک انحراف[۲۱] را نیز می‌آموزند. ستون چهارم از سمت چپ، نتیجه اعمال نکردن ورودی Exposure به شبکه است. این ورودی، مقادیر میزان روشنایی ثبت شده در هنگام عکاسی هستند که به عنوان ورودی به شبکه داده شده اند. این ورودی باعث می‌شود شبکه مفهوم روشن یا تاریک بودن تصویر را راحت‌تر متوجه شود. ستون سوم از سمت چپ، نتیجه اعمال نکردن بردار Appearance به شبکه است. این بردار را می‌توانید در شکل ۱۴ مشاهده کنید. وظیفه این بردار تغییردادن تصاویر مربوط به بلوک‌های همسایه برای یکسان سازی آن‌ها است.



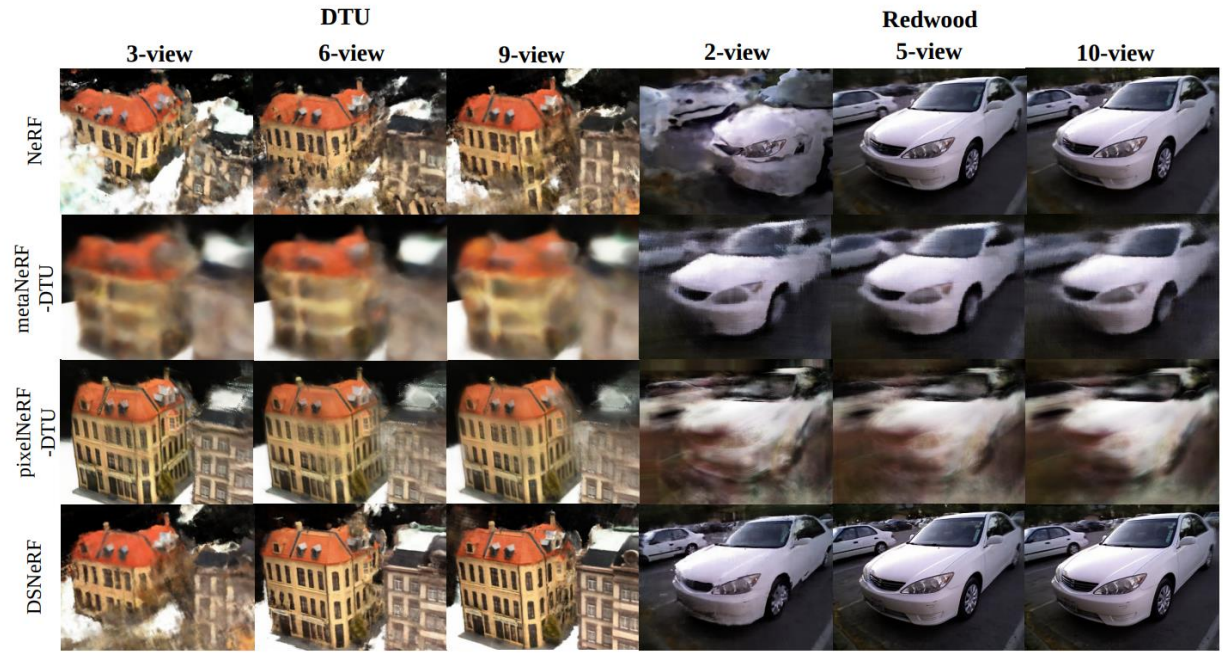
شکل ۱۵ - اثر بخش‌های مختلف این روش در خروجی نهایی

[21] offset

۳ـ۳. بهبود نتایج با تصاویر آموزش محدود (DS-NeRF[۲۲])

۱ـ۳ـ۳. مشکل نیاز به تصاویر آموزش زیاد

می‌دانیم هر چه تعداد تصاویر بیشتری از صحنه داشته باشیم، شبکه NeRF می‌تواند بهتر صحنه را بازنمایی کند. اما بسیاری از اوقات، دلیل استفاده از NeRF می‌تواند نداشتن تعداد کافی تصویر از یک صحنه باشد! به همین دلیل تعداد بسیار زیادی مقاله وجود دارد که برروی بهبود عملکرد NeRF با تعداد تصاویر آموزش محدود کار می‌کنند. یکی از این مقاله‌ها که عملکرد خوبی دارد مقاله‌ای به نام Depth-supervised NeRF: Fewer Views and Faster Training for Free [٤] است. همانطور که از نام مقاله پیداست، در این روش از یک ورودی اضافی(ورودی عمق) برای کمک کردن به روند آموزش شبکه NeRF استفاده می‌کنند. اگر به عنوان مقاله دقت کنید کلمه رایگان(free) را در آن می‌بینید. دلیل این نام گذاری این است که در روند آموزش NeRF، نیاز است با استفاده از روش‌هایی مانند SfM، زوایای دوربین موجود در هر تصویر را استخراج کنیم و به عنوان ورودی به شبکه بدهیم. روش‌هایی مانند SfM علاوه بر اطلاعات مربوط به دوربین موجود در هر تصویر، تعدادی نقطه کلیدی از درون صحنه نیز استخراج می‌کنند. این نقاط بسیار تنک هستند و قابل مقایسه با روش NeRF که یک میدان پیوسته از نقاط درون صحنه تولید می‌کند نیستند. اما با این حال این نقاط، اطلاعات بسیار مفیدی در خود دارند و به طور رایگان قبل از شروع آموزش، در اختیار ما هستند. یکی از اطلاعات مفیدی که این نقاط دارند، فاصله نقاط مختلف صحنه از دوربین است.

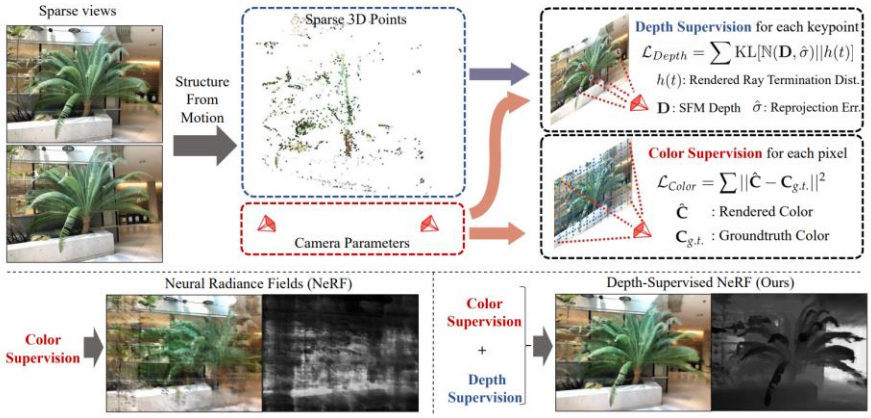


شکل ١٦ ـ تصاویر تولید شده توسط روش‌های مختلف با آموزش توسط تعداد مختلف داده

[22] Depth Supervised NeRF

۲ـ۳ـ۳. استفاده از عمق برای بهبود روند آموزش NeRF

با استفاده از عمق این نقاط، میتوان یک خطای جدید معرفی کرد. در این مقاله اسم این خطا را نظارت عمق[۲۳] گذاشته اند. رابطه این خطا را می‌توانید در شکل ۱۷ مشاهده کنید. هر موقع، فاصله انهدام[۲۴] یک پرتو در صحنه، با فاصله تخمین زده شده توسط SfM متفاوت باشد، خطای نظارت عمق زیاد خواهد شد و شبکه سعی خواهد کرد این خطا را کم کند.



شکل ۱۷ ـ نحوه عملکرد روش DS-NeRF

لازم به ذکر است که اطلاعات عمق مورد بحث، لزوما نباید از طریق SfM تولید شده باشند. به عنوان مثال جدول ۵ نتایج DS-NeRF را که عمق آن از طریق سنسورهای عمق در مجموعه داده تهیه شده است را نشان می‌دهد. در سطر آخر این اطلاعات عمق استفاده شده اند و نتیجه بهتری نسبت به تخمین عمق به کمک SfM بدست آمده است.

Redwood-3dscan [6]	PSNR↑			SSIM↑			LPIPS↓		
	2-view	5-view	10-view	2-view	5-view	10-view	2-view	5-view	10-view
NeRF	10.5	22.4	23.4	0.38	0.75	0.82	0.51	0.45	0.45
metaNeRF-DTU	14.3	14.6	15.1	0.37	0.39	0.40	0.76	0.76	0.75
pixelNeRF-DTU	12.7	12.9	12.8	0.43	0.47	0.50	0.76	0.75	0.70
MVSNeRF-DTU	-	17.1	17.1	-	0.54	0.53	-	0.63	0.63
finetuned	-	22.7	23.1	-	0.78	0.78	-	0.36	0.34
DS-NeRF	18.1	22.9	23.8	0.62	**0.78**	0.81	0.40	**0.34**	0.42
DS-NeRF w/ RGB-D	**20.3**	**23.4**	**23.9**	**0.73**	0.77	**0.84**	**0.36**	0.35	**0.28**

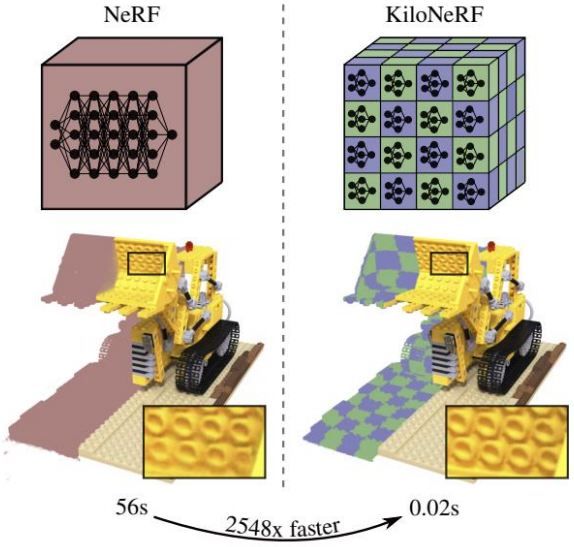
جدول ۴ – نتایج DS-NeRF بر روی مجموعه داده تصاویر RGB-D

[23] Depth Supervision

[24] Termination Distance

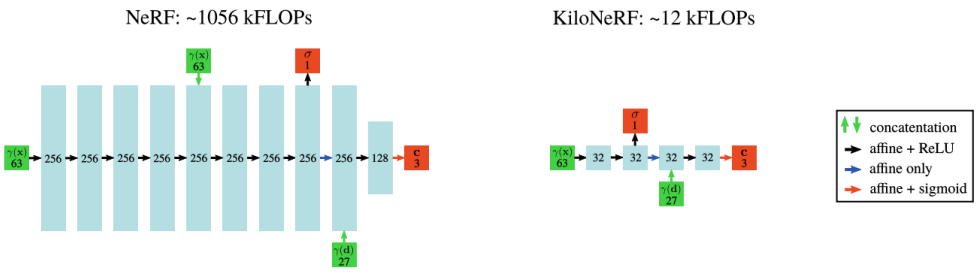
۴ـ۳. سرعت بخشی به NeRF (Kilo-NeRF)

۱ـ٤ـ۳. تقسیم یک شبکه MLP به هزاران MLP کوچکتر



شکل ۱۸ ـ مقایسه نحوه عملکرد و کارایی NeRF و Kilo-NeRF

همانطور که در بخش NeRF دیدیم، شبکه NeRF به دلیل نیاز به گرفتن میلیون‌ها خروجی از یک شبکه MLP نسبتا عمیق، بسیار کند است. تولید یک تصویر جدید با استفاده از NeRF حتی با پردازنده‌های گرافیکی مدرن حدود ۳۰ ثانیه طول می‌کشد. نویسنده‌های مقاله Kilo-NeRF[۵]، با قصد اینکه بتوان خروجی شبکه NeRF را به صورت بی‌درنگ نمایش داد، دست به طراحی شبکه ای جدید به اسم Kilo-NeRF زدند. همانطور که از اسم این شبکه پیدا است، این شبکه شامل هزاران شبکه NeRF دیگر است که هر کدام از این شبکه‌ها، وظیفه بازنمایی بخش کوچکی از صحنه را بر عهده دارند. این اتفاق باعث می‌شود هر شبکه، وظیفه بسیار سبکی داشته باشد که اجازه می‌دهد پیچیدگی هر شبکه را تا حد زیادی کاهش داد. در شکل ۱۹ می‌توان معماری هر MLP موجود در شبکه Kilo-NeRF را با MLP موجود در شبکه NeRF مقایسه کرد.



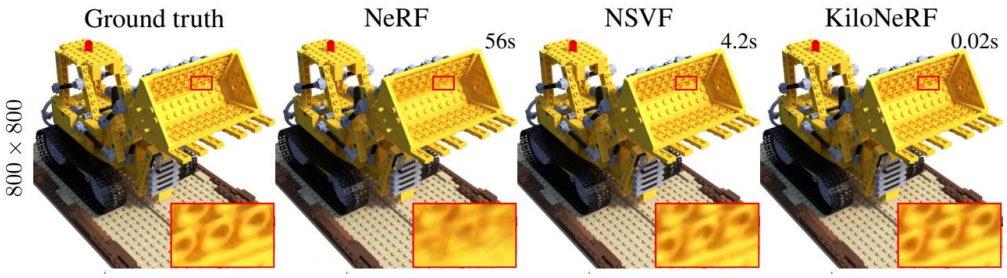
شکل ۱۹ ـ مقایسه معماری MLP موجود در NeRF و Kilo-NeRF

۲ـ٤ـ۳. آموزش شبکه

این شبکه، ابتدا صحنه را به صورت سه بعدی جدول بندی می‌کند. به هر سلول این جدول یک MLP کوچک اختصاص داده می‌شود. این MLP تنها موظف است یادبگیرد چگونه این سلول را بازنمایی کند. همانطورکه ممکن است حدس زده باشید، مستقل عمل کردن این شبکه‌ها نتیجه نامطلوبی به همراه دارد. به همین علت نویسنده‌های این مقاله از روش تقطیر[۲۵] برای آموزش این شبکه‌های کوچک استفاده می‌کنند. نحوه آموزش کلی طبق این روش به این شکل است که ابتدا یک شبکه NeRF معمولی آموزش می‌بیند. سپس، شبکه‌های کوچک سعی می‌کنند به این شبکه بزرگ به چشم معلم نگاه کنند و به ازای ورودی‌های مختلف، خروجی ای مشابه خروجی شبکه بزرگ ارائه دهند. به این شکل، مشکلاتی که به دلیل مستقل بودن این‌شبکه‌های کوچک بوجود آمده بود رفع می‌شود. پس از انجام شدن عمل تقطیر، وزن شبکه‌های کوچک به کمک داده‌های ورودی، یک دور تنظیم[۲۶] می‌شود. در جدول ۵ می‌توان این روش را با سایر روش‌های موجود مقایسه کرد. به سطر مربوط به زمان ترسیم تصاویر توجه کنید.

Resolution		BlendedMVS 768 × 576	Synthetic-NeRF 800 × 800	Synthetic-NSVF 800 × 800	Tanks & Temples 1920 × 1080
PSNR ↑	NeRF	27.29	31.01	31.55	28.32
	NSVF	26.90	**31.74**	**35.13**	28.40
	KiloNeRF	**27.39**	31.00	33.37	**28.41**
SSIM ↑	NeRF	0.91	**0.95**	0.95	0.90
	NSVF	0.90	**0.95**	**0.98**	0.90
	KiloNeRF	**0.92**	**0.95**	0.97	**0.91**
LPIPS ↓	NeRF	0.07	0.08	0.04	0.11
	NSVF	0.11	0.05	**0.01**	0.15
	KiloNeRF	**0.06**	**0.03**	0.02	**0.09**
Render time (milliseconds) ↓	NeRF	37266	56185	56185	182671
	NSVF	4398	4344	10497	15697
	KiloNeRF	**30**	**26**	**26**	**91**
Speedup over NeRF ↑	NSVF	8	13	5	12
	KiloNeRF	**1258**	**2165**	**2167**	**2002**

جدول ۵ — مقایسه عددی نتایج روش Kilo-NeRF و سایر روش‌ها



شکل ۲۰ ـ مقایسه نتایج روش Kilo-NeRF با سایر روش‌ها

[25] Distillation

[26] Finetune

۴. بحث و جمع‌بندی

در این گزارش، به معرفی شبکه NeRF و بررسی کاربردهای آن و آشنایی با نقاط قوت و ضعف آن پرداختیم. سپس سعی کردیم با بررسی روش‌های دیگر نشان دهیم تا چه حد ایده‌های نو می‌توانند حتی کارهای خوب را نیز بهبود بخشند. در طی دو سالی که این شبکه ایجاد گردیده است، تعداد بسیار زیادی مقاله برای بهبود دادن آن به چاپ رسیده اند. تعدادی از این مقالات، به بهبود سرعت و کارایی روش پرداخته اند و تعدادی سعی کرده اند با انجام تغییراتی، این روش‌را برای کاربردهای خاصی مناسب ترکنند. در تمام زمینه‌های بحث شده در این گزارش، همچنان جای پیشرفت زیادی وجود دارد. در آینده ای نزدیک باید شاهد افزایش سرعت آموزش این شبکه و تبدیل آن به یک روش استاندارد و مطمئن برای امر سنتز نما باشیم.

فهرست مراجع

[1] Mildenhall, Ben, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. "**Nerf**: Representing scenes as neural radiance fields for view synthesis." In *European conference on computer vision*, pp. 405-421. Springer, Cham, **2020**.

[2] Tancik, Matthew, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. "**Block-nerf**: Scalable large scene neural view synthesis." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8248-8258. **2022**.

[3] Barron, Jonathan T., Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. "**Mip-nerf**: A multiscale representation for anti-aliasing neural radiance fields." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5855-5864. **2021**.

[4] Deng, Kangle, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. "**Depth-supervised nerf**: Fewer views and faster training for free." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12882-12891. **2022**.

[5] Reiser, Christian, Songyou Peng, Yiyi Liao, and Andreas Geiger. "**Kilonerf**: Speeding up neural radiance fields with thousands of tiny mlps." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14335-14345. **2021**.

[6] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "**Attention is all you need.**" *Advances in neural information processing systems* 30. **2017**.

[7] Schonberger, Johannes L., and Jan-Michael Frahm. "**Structure-from-motion revisited**." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4104-4113. **2016**.

[8] Rahaman, Nasim, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. "**On the spectral bias of neural networks**." In *International Conference on Machine Learning*, pp. 5301-5310. PMLR, **2019**.

[9] Chen, Wenzheng, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. "**Learning to predict 3d objects with an interpolation-based differentiable renderer**." *Advances in Neural Information Processing Systems* 32. **2019**.

[10] Li, Tzu-Mao, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. "**Differentiable monte carlo ray tracing through edge sampling**." *ACM Transactions on Graphics (TOG)* 37, no. 6 . **2018**

[11] Sitzmann, Vincent, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. "**Deepvoxels**: Learning persistent 3d feature embeddings." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2437-2446. **2019**.