۱)

به نام خدا.

توی این ارائه میخوایم با مبحث دینویز کردن تصاویر دیجیتال آشنا بشیم و به طور خاص تمرکزمون رو روی دینویز کردن تصاویر Raytrace شده به کمک کامپیوتر بذاریم.

۲)

دینویزینگ تصاویر یکی از راههای بهبود تصویره که هم به عنوان یه مرحله پیش پردازش و هم به عنوان یه مرحله پس پردازش روی تصاویر اعمال میشه.

این تصاویر میتونن تصاویر پزشکی باشن، تصاویر نجومی باشن، یا میتونن مورد خاصی که توی این ارائه بهش میپردازیم یعنی تصاویر Ray trace شده باشن.

۳)

اگه با Raytracing آشنایی ندارید لازمه بدونید که یک روش برای تولید تصویر عه که بیش از ۲۰ ساله مبنای اصلی تولید تصاویر برای فیلم های هالیوودی و انیمیشن هاست‌. بنابراین صنایع خیلی بزرگی به این تصاویر نیاز دارن و هر گونه بهبود توی سرعت تولید این تصاویر یا کیفیتشون، نتایج مهمی توی دنیا داره.

۴)

متاسفانه وقت نمیشه مفصل راجع به نحوه تولید این تصاویر صحبت کنم پس به یه توضیح کوتاه بسنده میکنم.

توی دنیای واقعی یک منبع نور، به سمت اجسام مختلف بینهایت پرتو یا ray پرتاب میکنه.

این پرتو ها به اجسام برخورد میکنن، با توجه به ویژگی های سطح جسم رنگشون عوض میشه و در نهایت به چشم ما میرسن و ما رنگ نهایی این پرتو رو میبینیم.

توی این شکل هم مسیر یه پرتو فرضی رسم شده.

اما الگوریتم های raytracing که سعی میکنن این اتفاق رو شبیه سازی کنن، قاعدتاً نمیتونن بینهایت پرتو رو محاسبه کنن و به یک سمپل کوچیکی از تمام پرتو ها بسنده میکنن.

۵)

انتخاب این پارامترِ اندازه سمپل تاثیر مستقیمی روی میزان نویز تصاویر داره. همونطور که توی شکل زیر میبینید هر چی تعداد سمپل بر پیکسل بیشتر شده، تصاویر اصطلاحا noise free تر شدن و در عین حال

زمان مورد نیاز برای تولید این تصاویر هم خیلی زیادتر شده.

بخاطر همین موضوع، بدست آوردن کمترین میزان نویز با کمترین میزان سمپل یه حالت ایده آله …

۶)

اما آیا راهی هست که بتونیم با تعداد سمپل پایین، میزان نویز کمی داشته باشیم؟ جواب استفاده از دینویزر هاست.

همونطور که توی شکل میبینید، تصویر سمت چپ با ۸ سمپل بر پیکسل رندر شده و تصویر وسط نسخه دینویز شده همین تصویره که فقط ۴ ثانیه به زمان رندر اضافه کرده اما در مقایسه با تصویر سمت راست که ground truth هست و با ۵۱۲ سمپل و در زمان ۱۶ دقققیقه رندر شده تفاوت بصری چندانی نداره.

۷)

حالا که چی؟ و چرای denoising رو گفتیم، خوبه راجع به چگونگی انجام دادنش صحبت کنیم.

مثل خیلی دیگه از تسک های ماشین لرنینگ، دینویزینگ تصاویر هم راههای سنتی و آماری زیادی داره. تعدادی از این راههای قدیمی رو میتونید ببینید. اما حداقل در زمینه تصاویر Raytrace شده فقط بعد از پرطرفدار شدن Deep learning, CNN ها و به کارگیری Autoencoder ها بود که نتایج به سطح استاندارد های هالیوودی رسید و در صنعت قابل استفاده شدند.

۸)

راجع به Autoencoder ها اگه بخوایم بگیم، Autoencoder ها از دو بخش اصلی انکدر و دیکدر ساخته شدن و نحوه کارشون به این شکله که ورودی به واحد انکدر داده میشه و انکدر این داده ها رو به یک latent space میبره و در واقع یک representation میانی از ورودی ها میسازه. معمولا سعی میشه حجم latent space ها کوچیک انتخاب بشه تا مدل مجبور بشه داده های ورودی رو فشرده کنه و بهترین فیچر ها رو ازش استخراج کنه. بعد از این مرحله این representation میانی به دیکدر داده میشه و ازش خواسته میشه بدون دیدن داده های ورودی، اونها رو تولید کنه. معیار سنجش عملکرد این مدل ها reconstruction error یا تفاوت بین خروجی دیکدر و داده اصلی هست. توی زمینه تصاویر، معمولا از CNN ها برای عمل انکد و دیکد استفاده میشه. و اگه به پایین صفحه دقت کنید میبینید که اولین اشاره به Autoencoder ها به سال ۱۹۸۵ از جف هینتون و همکاراش برمیگرده ولی همون‌طور که گفتیم تا قبل از پرطرفدارشدن دیپ لرنینگ و CNNها این معماری برای دینویز کردن تصاویر استفاده نمیشد.

۹)

توی سال ۲۰۰۸ یه مقاله از yoshua bengio و همکاراش ارائه شد که توی این مقاله معماری به اسم denoising autoencoder رو معرفی کردند. این معماری بر خلاف اسمش اصلا هدفش دینویز کردن تصاویر یا کلا دینویز کردن نبود بلکه نحوه کار این معماری به این شکل بود که به representation میانی مقداری نویز اعمال میشد و از دیکدر خواسته میشد با دیدن این رپرزنتیشنِ نویزی، داده اصلی و بدون نویز رو تولید کنه که عملاً نوعی دینویز کردن ورودی بود.

اهداف اصلی نویسنده های این مقاله این بود که با این کار مدل مجبور بشه representation های robust تری از داده اصلی تولید کنه، از overfitting جلوگیری بشه و مدل به یک تابع همانی ساده تبدیل نشه.

اما به هر حال این ایده باعث تولید denoiser های امروزی شد که نحوه عملکردشون به این صورته که داده نویزی به انکدر داده میشه، و از دیکدر توقع میره نسخه بدون نویز داده ورودی رو تولید کنه.

۱۰) حالا لازمه بگیم چرا داریم به طور خاص در مورد تصاویر raytrace شده صحبت میکنیم این روش هایی که گفتیم به نظر میرسه روی هر نوع تصویری جواب بدند.

یکی از دلایل انتخابمون اینه که در حین پروسه تولید این نوع تصاویر، ما به تمام محیط دسترسی کامل داریم و میتونیم انواع اطلاعاتی که از محیط نیاز داریم رو داشته باشیم. این اطلاعات اضافه که بهشون auxiliary pass یا auxiliary input هم میگن، انواع مختلفی ازشون موجوده که تعدادیشونو میتونید سمت راست اسلاید ببینید. این داده های اضافه شبیه تصاویر RGBD هستن که علاوه بر رنگ پیکسل ها، عمق اونا رو هم در اختیار دارن و با کمک اونها میتونن فیچر های خیلی قوی تری تولید کنن. اما هر چی داده ورودی پیچیده تر باشه به داده آموزش بیشتری نیاز داریم که این ما رو دقیقا به فایده بعدی تصاویر raytrace شده میرسونه.

تولید داده آموزش برای کار با این نوع تصاویر خیلی آسون تر از زمینه های دیگه س.

مزیت بعدی این نوع تصاویر قابل شخصی سازی بودن داده آموزشه. من چند وقت پیش یه مقاله دیدم که توش تصاویر روز رو می‌گرفتن و نسخه شب همون تصویر رو تولید میکردن. هدف نویسنده های این مقاله این بود که چون برای آموزش اتومبیل های خودران اکثر ویدیوهای موجود در روز ضبط شده بودند اتومبیل ها در شب رفتار ضعیف تری داشتن. و این ها می‌خواستن با این کار کمبود داده از حالت شب رو جبران کنن. اگه در مورد تصاویر raytrace شده همچین نیازی داشته باشید خیلی راحتتر و با تغییر دادن چند پارامتر و زدن چند دکمه این نسخه از تصاویر تولید میشن.

۱۱)

توی اسلاید قبل به داده های کمکی اشاره کردیم.

اتفاقا معروف ترین مدل های امروزی که برای دینویز کردن تصاویر raytrace شده تولید شدن هم از این نوع داده ها به خوبی استفاده میکنن و توی بعضی شرایط که جلوتر میبینیم وجود این داده های کمکی خیلی هم ضروریه.

معروف ترین ابزار های موجود برای این کار هم

Open image denoise az intel

Va optix denoiser az nvidia

هستن که جفتشون روش های بر پایه autoencoder هستند و از اون داده های کمکی هم استفاده میکنن.

۱۲)

توی اسلایدهای پایانی این ارائه تعدادی از خروجی های open image denoise رو با هم میبینیم.

این تصویر رو توی اسلاید های اول ارائه هم به عنوان مثالی از دینویز کردن تصاویر دیدیم اما این تصویر در واقع خروجی openimagedenoise بود که با یه مقایسه ساده میبینیم با استفاده ازش یه بهبود حدود ۵۰ برابری روی سرعت تولید تصاویر داریم.

۱۳)

توی این مثال روی تخته سیاهی که توی تصویر بود زوم کردم تا کاربرد داده های کمکی رو با هم ببینیم.

تصویر سمت چپ تصویر نویزی ای هست که با ۴ سمپل بر پیکسل تولید شده ، تصویر وسطی نسخه دینویز شده همون تصویره که فقط از داده ی RGB برای دینویز کردن استفاده کرده.

نتیجه به نسبت داده ورودی خیلی بهتر شده اما اگه خوب دقت کنید قسمت چرخ دنده های بالای لوگو، اون اعداد ۳ و نقطه های امیرکبیر یه مقدار مات شدن که دلیلش وجود نویز خیلی زیاد توی تصویر ورودی بوده.

اما اگه تصویر سمت راست رو ببینید که نسخه دینویز شده با داده RGB به همراه یک داده اضافی به اسم Albedo عه می‌بینید که تمام مشکلاتی که به چشم میومد به طور کامل رفع شدن و فقط ۶ ثانیه به زمان تولید تصویر اضافه شده.

۱۴)

مطالب ارائه تموم شد و فکر میکنم از لیمیت ۱۰ دقیقه هم یه مقدار عبور کردم بخاطر همین خیلی سریع یه جمع بندی روی اراعه داشته باشیم و ببینیم چه چیزایی یادگرفتیم.

با raytracer ها آشنا شدیم و فهمیدیم چرا تصاویر ری تریس شده نویزی هستن.

فهمیدیم تصاویر raytrace شده مهم هستند و در نتیجه اهمیت دینویز کردنشون هم زیاده.

در مورد اتو انکدر ها به عنوان بهترین روش برای دینویز کردن تصاویر اشنا شدیم.

تفاوت و مزیت های تصاویر raytrace شده رو نسبت به سایر تصاویر دیدیم. و در نهایت مثال هایی از عملکرد این دینویزر ها و کاربرد داده های کمکی رو بررسی کردیم.

خیلی ممنون که گوش دادید.