گزارش مقاله

In Defense of Classical Image Processing: Fast Depth Completion on the CPU [1]

نویسنده گزارش:

محمدصادق آبادی‌جو [۱] ــ ۹۹۱۴۱۹۰۰۱

[۱] دانشجوی کارشناسی ارشد مهندسی کامپیوتر

گرایش هوش‌مصنوعی دانشگاه صنعتی مالک اشتر

In Defense of Classical Image Processing: Fast Depth Completion on the CPU

Jason Ku, Ali Harakeh, and Steven L. Waslander

Mechanical and Mechatronics Engineering Department
University Of Waterloo
Waterloo, ON, Canada

jason.ku@uwaterloo.ca, www.aharakeh.com, stevenw@uwaterloo.ca

***Abstract*—With the rise of data driven deep neural networks as a realization of universal function approximators, most research on computer vision problems has moved away from hand crafted classical image processing algorithms. This paper shows that with a well designed algorithm, we are capable of outperforming neural network based methods on the task of**

Many different approaches have been proposed for depth completion. These approaches range from simple bilateral upsampling based algorithms [8] to end-to-end deep learning based ones [9]. The latter are very attractive as they require minimal human design decisions due to their data driven

چکیده :

محیط فیزیکی اطراف ما، یک محیط سه‌بُعدی است، محیطی که هر جسم موجود در آن، دارای طول، عرض و ارتفاع (عمق) می باشد. زمانی که به این محیط نگاه می کنیم و یا توسط دوربین های معمولی از آن عکس می گیریم، یک محیط سه بُعدی بر روی یک صفحه دوبُعدی نگاشت می شود. این کار باعث از دست دادن بُعد سوم یا همان عمق می شود. این بُعد حاوی اطلاعات بسیار مهمی جهت تحلیل و بررسی موقعیت و ویژگی های اشیاء موجود در تصویر می باشد. انسان با استفاده از تکنیک‌های سه‌بُعدی ساز مغز و داشتن قابلیت دید دو چشمی ، محیط را به شکل سه‌بُعدی درک می‌کند اما زمانی که از محیط عکس می گیریم باید سعی کنیم در کامپیوتر، بُعد سوم صحنه را توسط تکنیک های موجود بازیابی کنیم. تشخیص عمق تصویر، مسئله ای بسیار مهم و اساسی در بحث بینائی ماشین و هوش مصنوعی می باشد که هنوز راه حل قطعی برای آن پیدا نشده است. هدف مقاله [1]، بدست آوردن عمق یک صحنه با توجه داده‌های لیدار تنک عمق از طریق روش‌های سنتی پردازش تصویر می باشد.

در این گزارش مقاله [1] مورد بررسی قرار می‌گیرد در این مقاله بدون استفاده از الگوریتم های یادگیری عمیق و تنها با استفاده از تکنیک های سنتی پردازش تصویر به ارائه یک الگوریتم تکمیل عمق که از زیرشاخه های روش تشخیص عمق می باشد از روی داده های تنک عمق می‌پردازد. این الگوریتم نتایج بسیار رقابتی در مقایسه با الگوریتم های یادگیری عمیق ارائه

داده است، و در زمان ارائه رتبه اول بنچمارک KITTI به این الگوریتم تعلق می‌گیرد. و همچنین این الگوریتم با مقدار ۱۲۸۸ میلیمتر در شاخص RMSE جز بهترین ها بوده است. اما پس از آن مدل های جدیدتری همانند PENet (RMSE = 730.8 mm)توانسته‌اند از این روش عملکرد بهتری را نشان دهند.

فهرست مطالب

فصل چهارم : تشریح پیاده سازی

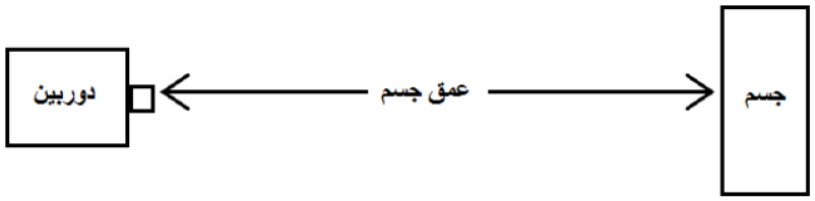
فصل پنجم : بحث و نتیجه گیری

فصل اول : مقدمه

۱-۱ـ پیشگفتار

محیط فیزیکی اطراف ما، یک محیط سه می باشد محیطی که هر جسم موجود در آن، دارای طول، عرض و عمق است. در واقع ما هر روز دیک محیط سه بعدی در حرکت هستیم. در اینجا دو سوال مهم مطرح می شود. اول آنکه مغز انسان چطور قادر به درک محیط محیط سه بعدی اطراف خود می باشد؟ سوال دوم این است که انسان چطور و با استفاده از چه تکنیک هایی می تواند از یک عکس دوبعدی ، بعد عمق را استخراج کند و ماشین که فاقد هوشمندی می باشد را قادر به درک عمق کند.

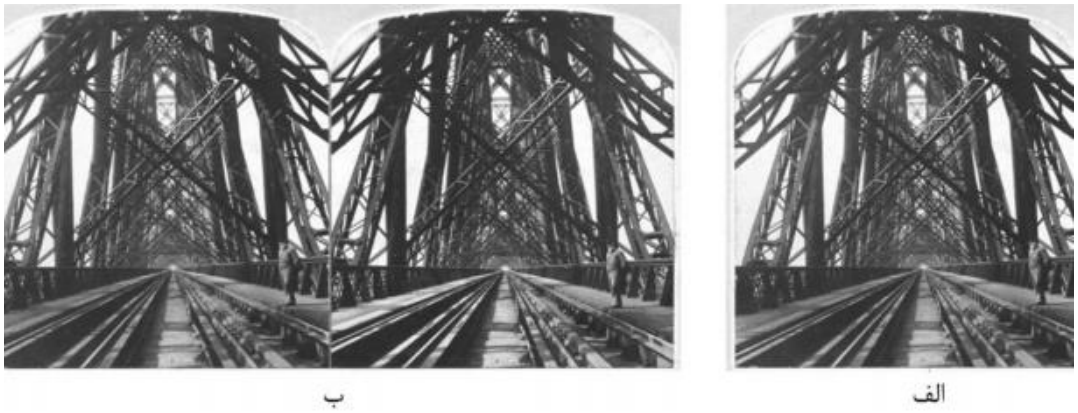
پیش از پاسخ به این دو سوال ، باید عمق شی را تعریف کنیم : همانطور که در شکل یک دیده می شود، به فاصله بین سطح قابل رویت جسم تا چشم انسان یا لنز دوربین عمق گفته می شود.



شکل شماره ۱ – مفهوم عمق شی

جهت روشم شدن موضوع، ابتدا سوال اول را بررسی می کنیم، همانطور که می دانیم چشم های هر انسان در حدود ۵ الی ۶ سانتی‌متر از یکدیگر فاصله دارند، بنابراین به محیط اطراف از دو زاویه افق کمی متفاوت نگاه می کنند. همچنان که ما به اطراف نگاه می کنیم، یک تصویر دوبعدی از محیط اطراف روی شبکیه چشم نگاشت می شود و شبکیه آن را ضبط و برای مغز ارسال می کند. مغز تصوریر دوبعدی را به صورت هوشمند و مطابق با تکنیک های سه بعدی ساز، طی فرآیندی ادغام و محیط اطراف را به شکل سه بعدی درک می کند. بنابراین انسان دارای قابلیت تشخیص عمق[1] می باشد. با توجه به توضیح بیان شده، به توانایی دیداری که منجر به درک حالت سه بعدی محیط پیرامون می شود، تشخیص عمق می گویند، توانایی که به ما اجازه می دهد اجسام دور را از اجسام نزدیک تشخیص دهیم و به راحتی درباره فواصل اشیا قضاوت کنیم. انسان به دلیل داشتن قابلیت دید دو چشمی، محیط اطراف خود را سه بعدی می بیند. اما جالب است بدانید که افرادی که دارای دید یک چشمی هستند نیز می توانند محیط اطراف را به شکل سه بعدی درک کنند. شکل ۲ تصویر حاصل از دید یک چشمی و دو چشمی را نشان می‌دهد.

[1] - Depth Estimation



شکل ۲ – الف) دید یک چشمی ب) دید دو چشمی

حال باید به سوال دوم پاسخ داد، زمانی که از محیط عکس میگیریم در حقیقت یک محیط سه بعدی را بر روی یک صفحع دوبعدی نگاشت کرده ایم، این کار باعث از دست دادن بعد سوم یا همان عمق می شود. همانطور که می دانیم یک تصویر دوبعدی، تنها دارای طول و عرض می باشد و از نظر فنی عمق ندارد. این بعد حاوی اطلاعات بسیار مهمی جهت تحلیل و بررسی موقعیت و ویژگی های اشیا موجود در تصویر می باشد و باعث می شود بتوان اجسام دور را از اجسام نزدیک تشخیص داد و به راحتی درباره فواصل اشیا قضاوت کرد. زمانی‌که انسان به یک تصویر که توسط دوربین تهیه شده است، نگاه می کند به طور ناخودآگاه و با استفاده از تکنیک های سه بعدی ساز مغز، تصویر را به شکل سه بعدی درک می کند اما در حقیقت در این تصویر همه چیز با یک فاصله ، از نگاه بیننده به نظر می‌رسد. بنابراین، برای آنکه سیستم کامپیوتری هم بتواند مانند انسان با توجه به اطلاعاتی که در تصویر وجود دارد، حالت سه بعدی تصویر را درک کند، با سعی کنیم این بعد (عمق) و حالت سه بعدی صحنه مورد نظر را توسط تکنیک های هوش مصنوعی موجود بازیابی کنیم [2].

۱-۲ـ ویژگی های تشخیص عمق

زمانیکه به یک منظره نگاه می‌کنیم، اطلاعاتی راجع به اینکه چه اجسامی هستند و کجا هستند، بدست می آوریم. درک مکان اجسام در هر صحنه تشخیص فضا نام دارد به دو گروه مسیر و جهت حرکت و فاصله تقسیم می شود. تخمین عمق، به اطلاعات دیداری ای که از چشم به مغز فرستاده می شود و نحوه پردازش آن اطلاعات توسط مغز، بستگی دارد [3]. با توجه به اینکه بیننده از چه ویژگی هایی جهت تخمین عمق استفاده می کند، این ویژگی ها به دو گروه عمده تقسیم می شوند :

۱-۲-۱ـ ویژگی های دوچشمی :

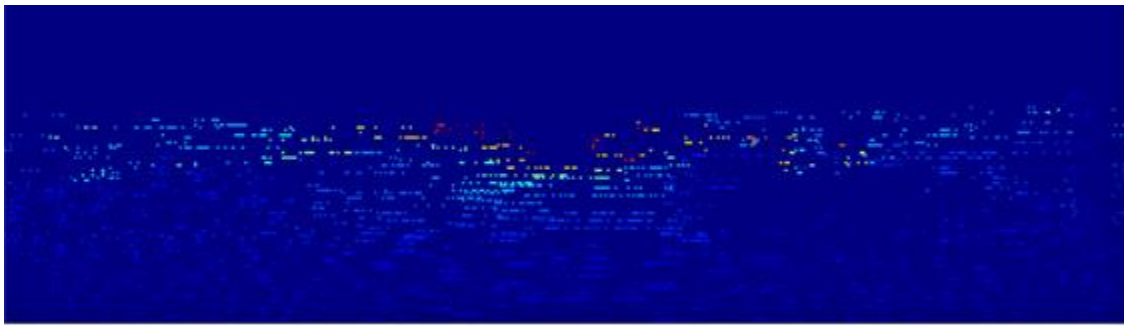
زمانی می‌توان جهت استخراج عمق یک صحنه از این ویژگی استفاده کرد که آن صحنه توسط دو دوربین تصویربرداری شده باشد. ناهمخوانی دیداری و همگرائی از جمله ویژگی های این گروه هستند.

۲-۲-۱ـ ویژگی های یک چشمی :

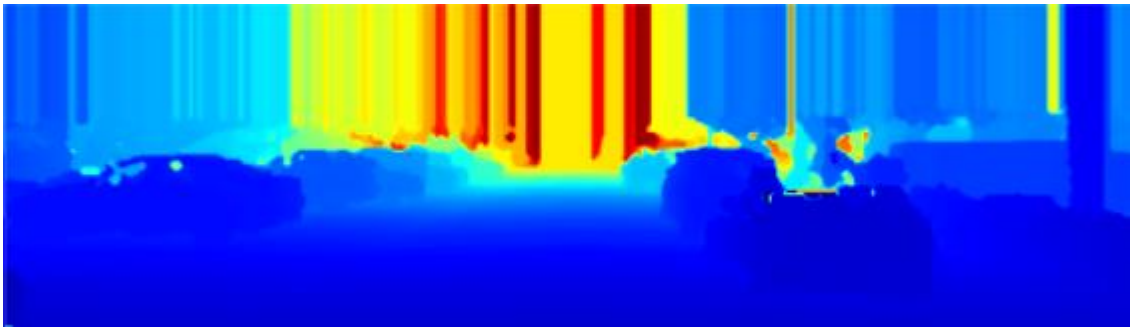
این ویژگی ها زمانی برای ما مفید هستند که صحنه مورد نظر توسط یک دوربین تصویربرداری شده باشد. ویژگی های تصویری و ویژگی های حرکتی بعنوان دو دسته از ویژگی های عمده این گروه شناخته می شوند.

۳-۱ـ تکمیل عمق[1]

تکمیل عمق یکی از زیر شاخه های تشخیص عمق می باشد که با استفاده از نقشه عمق تنک تصاویر[2]، که عموما از تصاویر لیدار استخراج می شوند به پیش بینی عمق یک تصویر می پردازد.تکمیل عمق از یک مجموعه پراکنده از مقادیر در نقشه عمق تنک تصاویر شروع می شود و عمق ناشناخته را برای پیکسل های باقی مانده که مقدار تهی دارند،تخمین می زند. بطورکلی تکمیل عمق تصویر را یک روش درونیابی برای بازیابی پیکسل تهی در نظر می گیرند با این روش نهایتا عمق تصاویر بدست می آید در شکل ۳، یک نمونه از نقشه عمق تنک تصاویر ارائه شده است و در شکل ۴ ، تصویری دیده می شود که با استفاده از فرایند تکمیل عمق مقاله [1] ارائه شده است.



شکل ۳ – نقشه تنک عمیق رنگ آمیزی شده با Opencv(برگرفته از دیتاست تکمیل عمق بنچمارک KITTI)



شکل ۴ـ تصویری که فرآیند تکمیل عمق مقاله مورد گزارش روی آن اجرا شده است

1- Depth Completion
2 - Sparse depth map

الگوریتم های تکمیل عمق تصاویر را می توان به دسته هدایت شده و هدایت نشده تقسیم کرد در روش های متعلق به دسته هدایت شده برای تکمیل عمق به تصاویر رنگی وابسته هستند در حالیکه روش های هدایت نشده تنها از نقشه های عمق تنک برای تکمیل عمق استفاده می کنند.

۱-۴ـ کاربردهای تشخیص عمق

تشخیص عمق از جمله مهم ترین مسائل مطرح در بینائی ماشین می باشد و در زمینه های صنعت، رباتیک، تجهیزات نظامی، بازسازی ساختار سه بعدی، بازسازی مدل سه بعدی چهره با توجه به یک تصویر دو بعدی از آن، بینایی سه بعدی، درک عمومی صحنه،... کاربرد فراوان دارد [4]. تشخیص عمق در مواردی همچون بازیابی پارامترهای سه بعدی حرکت ، کالیبراسیون دوربین، واسط های کامپیوتری ادراکی و فرآیندهای تشخیص سه بعدی نیز دارای کاربرد می باشد.

۱-۵ـ روش پیشنهادی مقاله [1] مورد گزارش

در مقاله [1] یک الگوریتم حریصانه ارائه شده است که در هشت گام اصلی از نقشه های تنک عمق ، عمق تصاویر طبیعی را بدست می آورد. در گام اول این الگوریتم ابتدا مقادیر نقشه های تنک بین عدد 0 تا 100 قرار می گیرند سپس در گام دوم با یک فیلتر مخصوص عملیات Dilation روی نقشه صورت می پذیرد تا پیکسل با مقدار صفر با مقدار نزدیک ترین پیکسل برابر گردد بعد از آن در گام سوم با استفاده از یک کرنل طراحی شده توسط نویسندگان مقاله عمل closing انجام خواهد شد این عمل منجر به بهبود لبه اشیا تصویر خواهد شد در گام بعدی مشابه گام قبل یک کرنل بزرگ تر closing به تصویر اعمال می شود تا حفره هایی با سایز medium را در تصویر بپوشاند. در گام پنجم با استفاده از مکانیزم گسترش مقدار مرتفع ترین پیکسل ها تا انتهای تصویر گسترش پیدا می کند در گام ششم با استفاده از یک فیلتر بزرگ تمامی پیکسل های تهی مقداردهی خواهند شد و نهایتا در گام هفتم یک فیلتر گوسی، میانه یا هردو به تصویر اعمال میشوند تا سطوح بدست آمده عمق هموارتر شوند و سپس در گام انتهایی مقادیر حاصل به مقادیر اصلی عمق در فاصله مشخص تبدیل خواهند شد.

۱-۶ـ ساختار گزارش

در این فصل دوم این گزارش ابتدا در مورد روش های مشابه بحث می شود در این فصل، یک مقاله با عنوان مشابه نیز شرح داده خواهد شد. همچنین در فصل سوم سامانه ارائه شده توسط مقاله مورد بررسی قرار می گیرد و اشکال های موجود در این سامانه مورد بررسی قرار خواهند گرفت. در فصل چهارم گزارش به شرح پیاده سازی سامانه پیشنهادی می پردازیم و همچنین در فصل پنجم ، نتایج حاصل از پیاده سازی با نتایج مقاله مورد قیاس قرار می گیرد نهایتا در فصل ششم به جمع مطالب ارائه داده شده خواهیم پرداخت.

فصل دوم : پیشینه پژوهش

۲-۱ـ پیشگفتار

امروزه تکمیل عمق در حوزه های مختلفی از جمله دید استریو، شار نوری و بازسازی سه بعدی تصویر دارای کاربردهای فراوانی می باشد و در اغلب مواقع فرآیند تکمیل عمق با استفاده از دادگان لیدار انجام می شود. در ادامه بخشی از پژوهش های موجود در حوزه تخمین عمق و در ادامه در حوزه تکمیل عمق را مورد بررسی قرار خواهیم داد.

۲-۲ـ پژوهش های مشابه

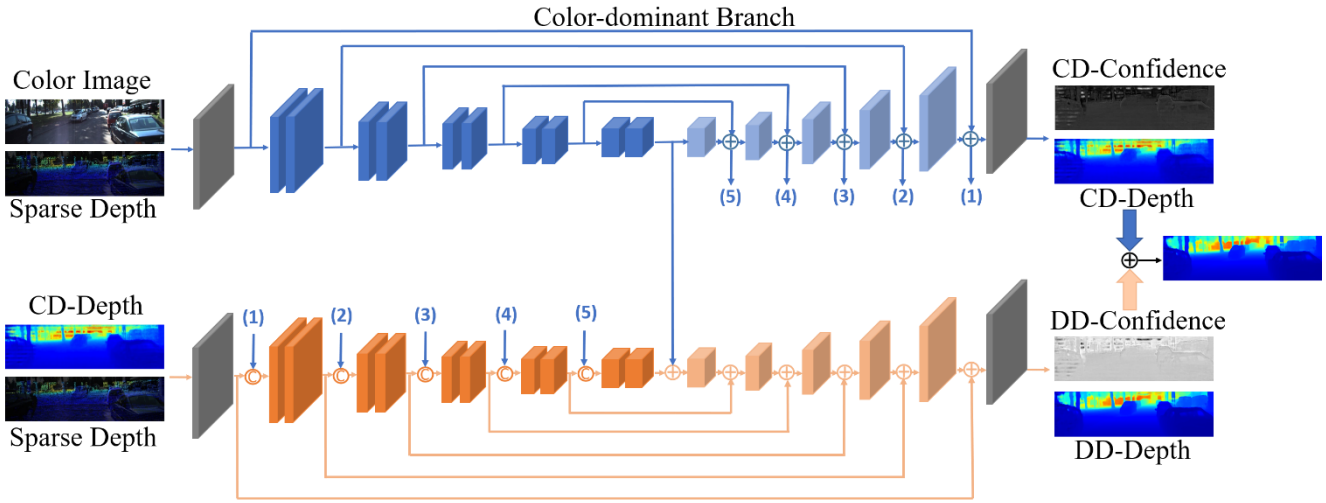
در مقاله [5] نویسندگان با ساختار عمومی تصویر و اعمال تبدیل فوریه به صورت عمومی و تبدیل ویولت به صورت محلی، عمق میانگین تصویر را بدست آورده اند. مقاله [5] عدم نیاز به بخش بندی تصویر را به عنوان مزیت و برتری کار خود عنوان کرده است. روشی که در مقاله [6] توسط نوسینده بکار گرفته شده است بدین ترتیب است که به بخش‌بندی تصویر می پردازد. ابتدا نواحی‌ای مثل آسمان و زمین که به ترتیب دارای بیشترین و کمترین عمق هستند را استخراج می کند و این بخشبندی را در مورد نواحی بین این دو قسمت اجرا می کند و سپس عمق نواحی موجود را با توجه به این دو نشانه تخمین می زند. در این روش از ویژگی های مکانی و شکلی، ویژگی رنگ، ویژگی بافت استفاده شده است. از جمله مزایای این روش می توان به عدم نیاز به وجود فاز یادگیری ، تشخیص عمق به صورت patch-base و نه به صورت point-base، اندازه نواحی متغیر است و الزاما با هم برابر نیست، اشاره کرد. تنها مشکل این روش وجود الزامی نشانه هایی مثل آسمان و زمین در تصاویر می باشد.

با معرفی و فراگیری استفاده از یادگیری عمیق در بینایی کامپیوتر، به تبع پژوهشگران حوزه تخمین و تکمیل عمق نیز به سمت استفاده از این روش ها برای استفاده پیش بینی عمق تصویر حرکت کرده اند در مقاله های [7] و [8] روش هایی مبتی بر یادگیری عمیق ارائه شده است که دارای دقت بسیار مناسبی برای این حوزه هستند منتها این روش ها نیاز به دادگان بسیار گسترده دارند. در ادامه به تشریح روش PENet [9] می پردازیم این روش یکی از بروزترین روش های حوزه تکمیل عمق می باشد و همچنین در بنچمارک KITTI Depth completion از رتبه های برتر می باشد و در بخش پنجم نتایج آن با نتایج مقاله مورد مقایسه قرار خواهند گرفت. پس از معرفی و تشریح مقاله [9] به تشریح روش Monodepth [10] خواهیم پرداخت. این روش نیز جز قوی ترین روش های حوزه تخمین عمق می باشد.

۲-۲-۱ـ تشریح سامانه تخمین عمق PENet

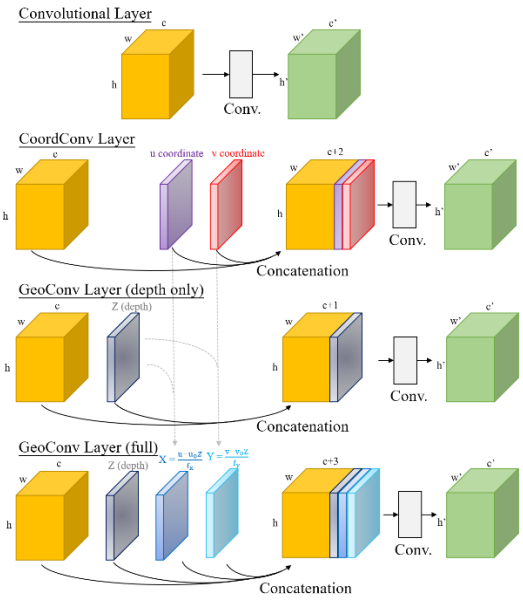
این روش یک معماری دو شاخه ای شامل شاخه های color dominant branch و depth dominant branch برای انجام فرآیند تکمیل عمق ارائه می دهد. در شاخه color dominant branch تصویر اصلی قرار می گیرد و نقشه تنک عمق تولید می شود در شاخه دوم که نتیجه ای مشابه روش مقاله مورد گزارش دارد ، با استفاده از نقشه تنک تولید شده در شاخه قبلی

به تکمیل و ارائه عمق تصاویر طبیعی می پردازد. عمق ارائه شده توسط این شبکه به مراتب نسبت به روش مقاله [1] دقیق تر می باشد. به علاوه در روش PENet یک لایه کانوولشن هندسی ارائه شده است این چارچوب کدگذاری هندسی تلفیقی از اشکال مختلف را در چندین فاز انجام می دهد تا نتایج کاملتر عمق بهتر حاصل شود. این چارچوب کدگذاری هندسی، تلفیقی از اشکال مختلف را در چندین فاز انجام می دهد تا نتایج کاملتر عمق بهتر حاصل شود.



شکل ۵ – معماری PENet

معماری دو شاخه به منظور بهره برداری کامل از اطلاعات غالب رنگ و عمق غالب از شاخه های مربوطه آنها و همجوشی دو روش موثر ساخته شده است.



شکل ۶ – معماری لایه کانوولوشن هندسی

در شکل ۶ ساختار دیده می شود که وظیفه رمزگشایی اطلاعات هندسی سه بعدی تصویر را بر عهده دارد. این چارچوب کدگذاری هندسی، تلفیقی از اشکال مختلف را در چندین فاز انجام می دهد تا نتایج کاملتر عمق بهتر حاصل شود. در فصل پنجم عملکرد این روش با عملکرد روش مقاله مورد گزارش مقایسه خواهد شد.

۲-۲-۲ـ سامانه تخمین عمق Monodepth

نویسندگان مقاله [10] روشی بسط داده‌اند که در آن شبکه های عمق در کنار شبکه‌های ژست قرارگرفته اند تا بدین وسیله عمق در یک فریم را پیش‌بینی می‌کنند. نویسندگان این مقاله برای پیش‌بینی عمق، معماری خود را با رشته‌ای از فریم‌ها آموزش می‌دهند و علاوه بر آن از چندین تابع زیان برای آموزش این دو شبکه استفاده می‌کنند. در این روش برای آموزش، به دیتاست برچسب‌دار نیازی نیست. در عوض در این روش از فریم‌های متوالی زمانی یک عکس برای آموزش استفاده می‌شود. علاوه بر این در این روش برای کمک به یادگیری محدود، از یک شبکه تخمین ژست استفاده می‌شود. مدل با تفاوت‌های میان تصویر ورودی و تصویری که از خروجی شبکه ژست و شبکه عمق بازسازی شده آموزش می‌بیند.اصلی‌ترین یافته‌های این مقاله عبارتند از:۱ -تکنیک ماسک‌کردن خودکار برای تمرکز بر روی پیکسل‌های مهم‌تر ۲ -اصلاح خطای فتومتریک بازسازی Photometric reconstruction errorبا استفاده ازنگاشت عمق ۳ـ تخمین چند مقیاسی عمق. در روشی که در این مقاله ارائه شده است از یک شبکه عمق و یک شبکه ژست استفاده می‌شود. شبکه عمق معماری انکودرـ دیکودر U-Net است. انکودر مدل ResNet می باشد که از پیش آموزش دیده است. دیکودر عمق مشابه مقاله [9] می باشد که در آن خروجی سیگموئید را به مقادیر عمق تبدیل می‌کند.

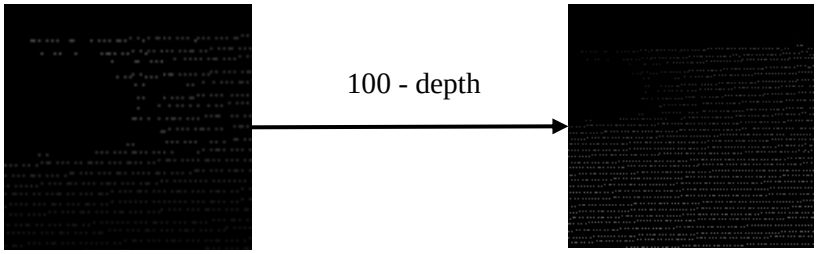
فصل سوم : سامانه پیشنهادی

۳-۱ـ پیشگفتار

در مقاله [1] یک الگوریتم حریصانه ارائه شده است که در هشت گام اصلی از نقشه های تنک عمق ، عمق تصاویر طبیعی را بدست می آورد. در گام اول این الگوریتم ابتدا مقادیر نقشه های تنک بین عدد 0 تا 100 قرار می گیرند سپس در گام دوم با یک فیلتر مخصوص عملیات Dilation روی نقشه صورت می پذیرد تا پیکسل با مقدار صفر با مقدار نزدیک ترین پیکسل برابر گردد بعد از آن در گام سوم با استفاده از یک کرنل طراحی شده توسط نویسندگان مقاله عمل closing انجام خواهد شد این عمل منجر به بهبود لبه اشیا تصویر خواهد شد درگام بعدی مشابه گام قبل یک کرنل بزرگ تر closing به تصویر اعمال می شود تا حفره هایی با سایز medium را در تصویر بپوشاند. در گام پنجم با استفاده از مکانیزم گسترش مقدار مرتفع ترین پیکسل ها تا انتهای تصویر گسترش پیدا می کند درگام ششم با استفاده از یک فیلتر بزرگ تمامی پیکسل های تهی مقداردهی خواهند شد و نهایتا در گام هفتم یک فیلتر گوسی، میانه یا هردو به تصویر اعمال میشوند تا سطوح بدست آمده عمق هموارتر شوند و سپس در گام انتهایی مقادیر حاصل به مقادیر اصلی عمق در فاصله مشخص تبدیل خواهند شد. این الگوریتم در زمان ارائه توانسته با ثبت مقدار ۱۲۸۸ برای شاخص RMSE ، از بهترین ها باشد. در ادامه این فصل گام های الگوریتم پیشنهادی مورد بررسی قرار می گیرد.

۳-۲ـ گام اول : معکوس کردن عمق

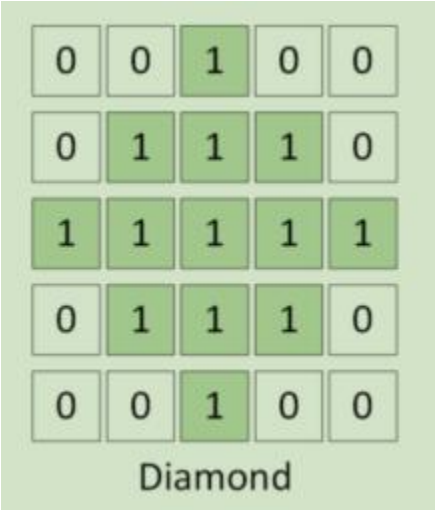
در دیتاست KITTI Raw depth map مقادیر بین ۰ تا ۸۰ متر قرار می گیرند . در نتیجه قسمت هایی که مقدار صحیح برای آن ها پیدا نشده با صفر مقداردهی خواهند شد این موضوع منجر می شود که با اعمال مستقیم عملگر dilation در گام بعدی ، منجر به از بین رفتن فواصل کوتاه بین اشیا خواهد شد لذا منجر به از بین رفتن لبه های اشیا می شود. برای جلوگیری از این موضوع مقادیر غیردهی عمق از ۱۰۰ کم می شوند تا فاصله بین مقادیر تهی و مقادیر بسیار بزرگ به حداقل برسد و در نتیجه مشکل فوق بوجود نخواهد آمد. به نوعی در این گام یک نرمالسازی صورت می پذیرد تا در گام های بعد عملیات مورد نظر بهترین نتیجه را به همراه داشته باشند.شکل



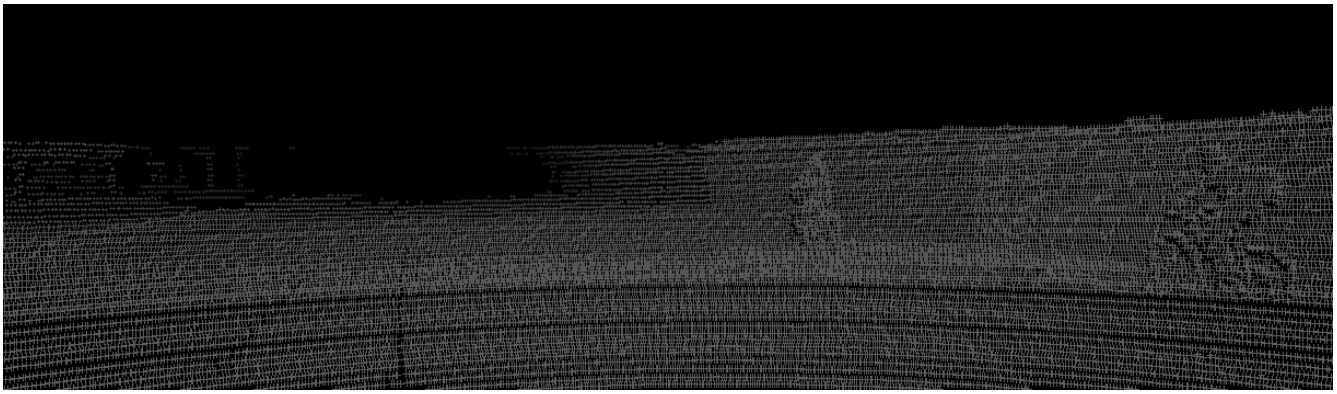
شکل ۷ـ نتیجه گام اول

۳-۳ـ گام دوم : پر کردن فضای خالی نزدیک پیکسل های موثر

در این گام پیکسل های نزدیک به پیکسل های دارای مقادیر موثر مقداردهی می شوند. پیکسل های نزدیک پیکسل های موثر با احتمال بسیار بالایی دارای عمق برابر با پیکسل های موثر هستند لذا نویسندگان این مقاله با اعمال سه فیلتر الماس ۳*۳ ، ۵*۵ و ۷*۷ برای عمق های دور ، متوسط و نزدیک به نتیجه دلخواه می رسند در پیاده سازی مقاله تغییراتی روی سایز و نوع فیلترها صورت پذیرفت منتها تاثیر چندانی برروی نتیجه کار دیده نشد. و حتی در مقایسه نهایی شاخص RMSE در بدترین حالت تنها ۴ واحد دچار تغییر گشت.



شکل ۸ – فیلتر الماس ۵*۵

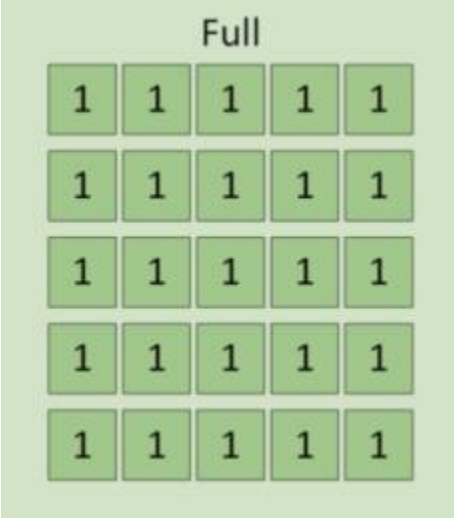


شکل ۹ – نتیجه اعمال فیلتر الماس ۵*۵

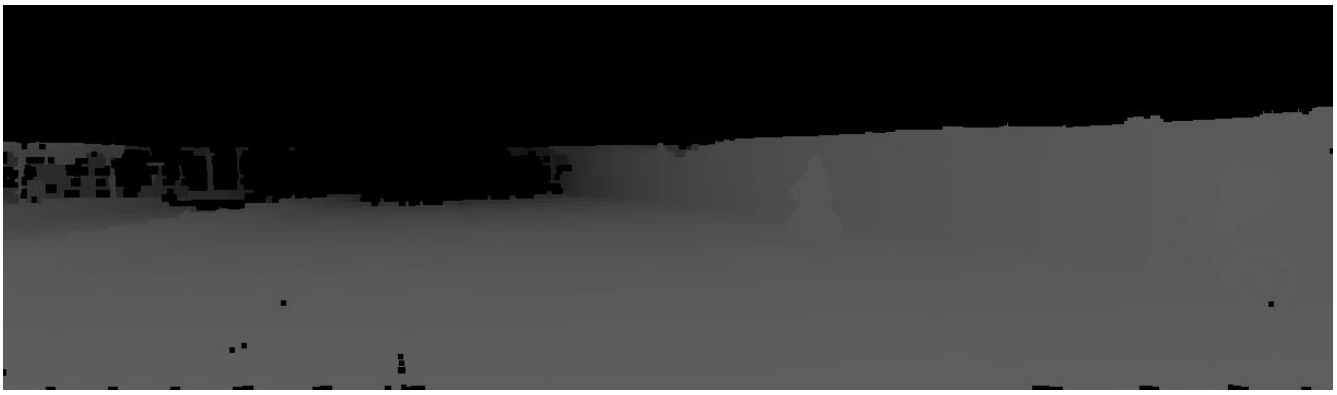
۴-۳ـ گام سوم : پرکردن فضاهای خالی بین پیکسل های موثر در نقشه عمق

در این مرحله با اعمال یک فیلتر ۵*۵ کامل (تمامی درایه های فیلتر دارای مقدار یک هستند) و پیکسل های جدا از هم به یکدیگر وصل می شوند و فضای بین آن ها پر می شود. در این مرحله میتوان از فیلترهای ۳*۳ و ۷*۷ هم استفاده کرد که

تقریبا حدود ۱۰۰ واحد مقدار RMSE را کاهش می دهند. در شکل ۱۱ نتیجه اعمال این فیلتر را می بینید. در نهایت پس از انجام عملیات dilation برای حذف نویز از تصویر یک فیلتر ۵*۵ میانه نیز به آن اعمال خواهد شد.



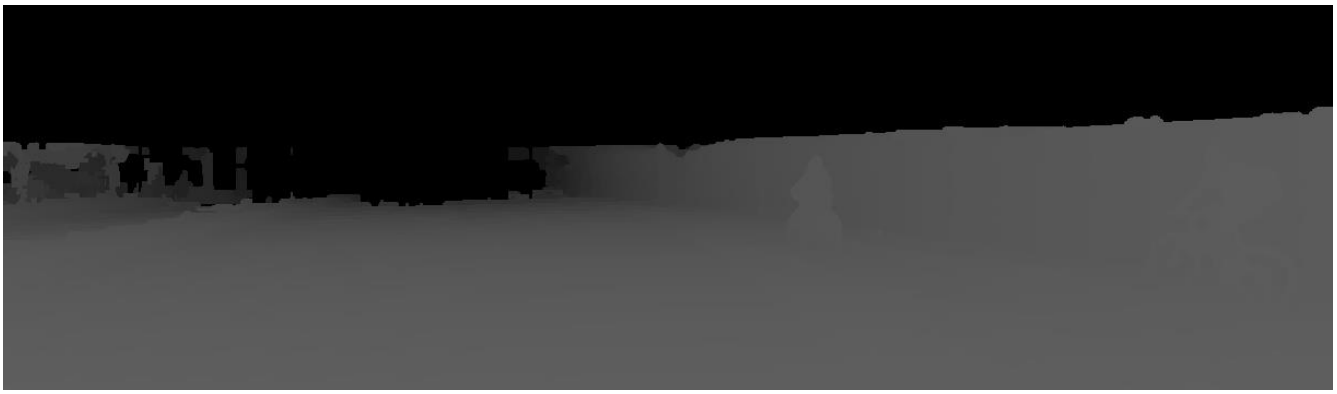
شکل ۱۰ – فیلتر کامل ۵*۵



شکل ۱۱ – نتیجه اعمال فیلتر کامل ۵*۵

۵-۳ـ گام چهارم : پر کردن حفره های خالی با سایز متوسط با مقدار پیکسل های مجاور

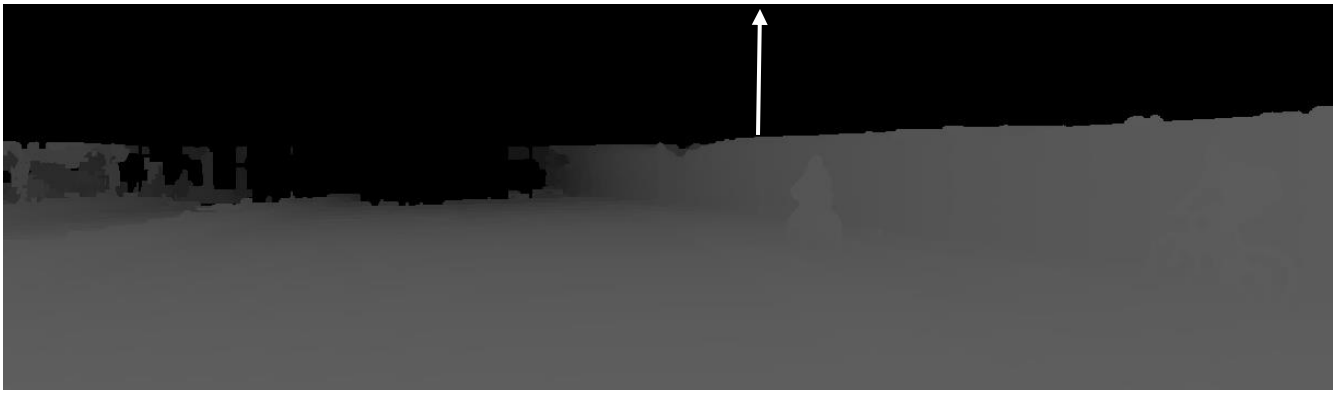
بعد از انجام گام سوم بعضی از قسمت های نقشه هنوز خالی باقی می مانند با اعمال یک فیلتر بزرگ تر برروی نقشه این پیکسل های خالی نیز با مقدایر پیکسل های اطراف خود مقداردهی میشوند. فیلتر استفاده شده در این قسمت توسط نویسندگان مقاله یک فیلتر ۷*۷ کامل می باشد. در این مرحله ابتدا یک فیلتر میانه به تصویر اعمال می شود سپس عملیات Dilation روی تصویر با فیلتر کامل اعمال می شود. نتیجه انجام این فرایند در شکل ۱۲ آورده شده است. در پیاده سازی از فیلترهای کامل ۹*۹ و ۵*۵ نیز استفاده شد که تغییر چندانی نسبت با حالت استاندارد ۷*۷ بوجود نیامد.



شکل ۱۲ – نتیجه گام چهارم

۶-۳ـ گام پنجم : گسترش مقدار نزدیک ترین پیکسل های هر ستون تصویر تا انتهای تصویر

در این مرحله از طریق برون یابی بزرگترین مقدار پیکسل های هر ستون تا انتهای تصویر امتداد پیدا میکنند همانطور که در شکل ۱۳ می بینید پیکسل های بزرگ تر هر ستون مقدارشان تا انتهای تصویر گسترش پیدا می کنند. یکی از فواید این کار این است که اگر دو شی هم را مسدود کرده باشند نزدیک ترین آن ها شناسایی شود.



شکل ۱۳ – نحوه گسترش

نتیجه حاصل از اعمال این فیلتر در شکل شماره ۱۴ آورده شده است.

شکل ۱۴ – نتیجه گام ۵

۳-۷ـ گام ششم : پرکردن حفره های بزرگ در نقشه

پس از اعمال گام پنجم حفره هایی در تصویر مشاهده می شود در شکل ۱۵ این حفره ها مشخص شده اند. حفره های مشخص شده در تصویر با اعمال Dialation با یک فیلتر کامل ۳۱*۳۱ از بین می روند نتیجه اعمال این فیلتر را در شکل ۱۶ می بینید.

شکل ۱۶ – حفره های داخل تصویر

شکل ۱۷ – نتیجه گام ششم و هفتم

۳-۸ـ گام هفتم : اعمال فیلتر Median و Gaussian بر تصویر

در این مرحله ابتدا فیلتر median به تصویر اعمال می شود که نویز تصویر را کاهش دهد سپس با اعمال یک فیلترگوسی به تصویر لبه های اشیا هموار تر می شوند مطابق گفته مقاله اعمال این دو فیلتر می تواند در تشخیص اشیا داخل تصویر موثرتر باشد. دو فیلتر استفاده شده در این گام دارای ابعاد ۵*۵ می باشند. شکل ۱۷ نتیجه اعمال این دو فیلتر را نیز در بردارد. در پیاده سازی صورت گرفته یک بار نیز فیلتر biliteral هم اعمال شد که نتیجه با اعمال فیلترگوسی تفاوت چندانی نداشت.

۳-۹ـ گام هشتم : برگرداندن مقادیر عمق به مقدار اصلی

در گام اول ما مقادیر عمق را از بیشتر عمق کم کردیم در این قسمت نیز مقادیر بدست آمده را دوباره از عمق اصلی کم میکنیم تا نتیجه نهایی بدست بیاید. شکل ۱۸ نتیجه نهایی گام هشتم را نشان میدهد.

شکل ۱۸ – گام هشتم

فصل چهارم : تشریح پیاده سازی

۴-۱ـ پیشگفتار

برای پیاده سازی الگوریتم فوق از زبان برنامه نویسی پایتون و منحصرا از کتابخانه opencv و numpy استفاده شده است ساختار برنامه ارائه شده دقیقا مطابق با الگوریتم انجام شده است یعنی گام به گام موارد بیان شده در الگوریتم مورد پیاده سازی قرار گرفته است برای پیاده سازی موارد dilation و closingاز کلاس morphology موجود در کتابخانه opencv استفاده می شود همچنین فیلترهای موجود نیز از این کتابخانه برداشته شده اند. در ادامه گام به گام به شرح پیاده سازی الگوریتم خواهیم پرداخت.

۴-۲ـ گام اول : معکوس کردن عمق

در این گام فقط نیاز است پیکسل های valid(پیکسل هایی که مقدار آن ها از ۰,۱ بزرگتر است منظور حذف پیکسل هایی می باشد که عمق تقریبا صفر دارند) را از مقدار ۱۰۰ یا هر بزرگترین عمقی کم کنیم. کد به شکل زیر می باشد .

```
s1_inverted_depths = np.copy(depths_in)
valid_pixels = (s1_inverted_depths > 0.1)
s1_inverted_depths[valid_pixels] = \
    max_depth - s1_inverted_depths[valid_pixels]
```

۴-۳ـ گام دوم : پر کردن فضای خالی نزدیک پیکسل های موثر

در این گام مطابق مقاله مورد گزارش سه فیلتر الماس باید تعریف کنیم ، فیلتر اول برای عمق زیاد، فیلتر دوم برای عمق متوسط و فیلتر سوم برای عمق کمتر مورد استفاده قرار می گیرد هرچه عمق زیادتر میشود طول ابعاد فیلتر نیز کاهش می یابد زیرا در عمق بیشتر نقاط نقشه تنک عمق متراکم تر هستند. در بخش ۱۰ـ۳ پیاده سازی فیلترها شرح داده خواهد شد. در ضمن برای انجام dilation رو تصویر از تابع dilate کتابخانه opencv استفاده شده است.

```
dilated_far = cv2.dilate(
    np.multiply(s1_inverted_depths, valid_pixels_far),
    dilation_kernel_far)
dilated_med = cv2.dilate(
    np.multiply(s1_inverted_depths, valid_pixels_med),
    dilation_kernel_med)
dilated_near = cv2.dilate(
    np.multiply(s1_inverted_depths, valid_pixels_near),
    dilation_kernel_near)

# Find valid pixels for each binned dilation
valid_pixels_near = (dilated_near > 0.1)
valid_pixels_med = (dilated_med > 0.1)
```

```
valid_pixels_far = (dilated_far > 0.1)

# Combine dilated versions, starting farthest to nearest
s2_dilated_depths = np.copy(s1_inverted_depths)
s2_dilated_depths[valid_pixels_far] = dilated_far[valid_pixels_far]
s2_dilated_depths[valid_pixels_med] = dilated_med[valid_pixels_med]
s2_dilated_depths[valid_pixels_near] = dilated_near[valid_pixels_near]

    dilation_kernel_near)
```

۴-۴ـ گام سوم : پرکردن فضاهای خالی بین پیکسل های موثر در نقشه عمق

برای پیاده سازی این قسمت یک فیلتر کامل ۵*۵ را با استفاده از تابع cv2.MORPH_CLOSE و همچنین بکارگیری آن در cv2.morphologyEx میتوانیم عملیات closing را انجام دهیم. سپس برای حذف نویزهای ایجاد شده یک فیلتر میانه ۵*۵ را به تصویر اعمال می کنیم. همچنین دقت شود که برخی از پیکسل های تصویر باز هم مقدار نمی گیرند که ممکن است بعد از اعمال فیلتر میانه مقادیر بسیار کوچکی دریافت کنند که میتوان با استفاده از یک valid_mask دوباره این مقادیر را تهی کرد وجود این مقادیر نزدیک صفر می تواند در محاسبات بعد باز هم ایجاد حفره کند.

```
s3_closed_depths = cv2.morphologyEx(
    s2_dilated_depths, cv2.MORPH_CLOSE, kernels.FULL_KERNEL_5)

s4_blurred_depths = np.copy(s3_closed_depths)
blurred = cv2.medianBlur(s3_closed_depths, 5)
valid_pixels = (s3_closed_depths > 0.1)
s4_blurred_depths[valid_pixels] = blurred[valid_pixels]
```

۴-۵ـ گام چهارم : پر کردن حفره های خالی با سایز متوسط با مقدار پیکسل های مجاور

در این گام ابتدا می بایست مقادیری که در گام قبلی تغییر نکرده اند را جدا کنیم سپس با انجام عملیات dilation با یک ماسک ۷*۷ به حفره ها را پر می کنیم در کد زیر ابتدا تصویر بصورت ستونی (هر پیکسل یک ستون در راستای عمود) اسکن می شود و مقادیری که در گام قبل تغییر کرده اند و به اصطلاح خفره نیستند مقدار False میگیرند با این عمل عملگر dilation فقط می تواند برروی حفره اثر کند و بقیه پیکسل ها را تغییر ندهد.

```
top_mask = np.ones(depths_in.shape, dtype=np.bool)
for pixel_col_idx in range(s4_blurred_depths.shape[1]):
    pixel_col = s4_blurred_depths[:, pixel_col_idx]
    top_pixel_row = np.argmax(pixel_col > 0.1)
    top_mask[0:top_pixel_row, pixel_col_idx] = False

valid_pixels = (s4_blurred_depths > 0.1)
empty_pixels = ~valid_pixels & top_mask

dilated = cv2.dilate(s4_blurred_depths, kernels.FULL_KERNEL_7)
s5_dilated_depths = np.copy(s4_blurred_depths)
s5_dilated_depths[empty_pixels] = dilated[empty_pixels]
```

۴-۶ـ گام پنجم : گسترش مقدار بزرگترین هر ستون (نزدیک ترین) پیکسل ها تا انتهای تصویر

برای پیاده سازی این قسمت ابتدا بزرگ ترین مقدار هر ستون را بدست می آوریم سپس مقدار پیکسل حاصل را از مرز ناحیه سیاه تا انتهای آن بسط می دهیم

```
s6_extended_depths = np.copy(s5_dilated_depths)
top_mask = np.ones(s5_dilated_depths.shape, dtype=np.bool)

top_row_pixels = np.argmax(s5_dilated_depths > 0.1, axis=0)
top_pixel_values = s5_dilated_depths[top_row_pixels,
                                     range(s5_dilated_depths.shape[1])]
for pixel_col_idx in range(s5_dilated_depths.shape[1]):
    s6_extended_depths[0:top_row_pixels[pixel_col_idx],
                       pixel_col_idx] = top_pixel_values[pixel_col_idx]
```

۴-۷ـ گام ششم : پرکردن حفره های بزرگ در نقشه

در این قسمت یک فیلتر کامل ۳۱*۳۱ را بر تصویر اعمال می کنیم.

```
s7_blurred_depths = np.copy(s6_extended_depths)
for i in range(6):
    empty_pixels = (s7_blurred_depths < 0.1) & top_mask
    dilated = cv2.dilate(s7_blurred_depths, kernels.FULL_KERNEL_31)
    s7_blurred_depths[empty_pixels] = dilated[empty_pixels]
```

۴-۸ـ گام هفتم : اعمال فیلتر Median و Gaussian بر تصویر

در این قسمت ابتدا با استفاده از کتابخانه opencv دو فیلتر ۵*۵ میانه و گوسی را بر تصویر اعمال می کنیم.

```
s7_blurred_depths = np.copy(s6_extended_depths)
for i in range(6):
    empty_pixels = (s7_blurred_depths < 0.1) & top_mask
    dilated = cv2.dilate(s7_blurred_depths, kernels.FULL_KERNEL_31)
    s7_blurred_depths[empty_pixels] = dilated[empty_pixels]
blurred = cv2.medianBlur(s7_blurred_depths, 5)
valid_pixels = (s7_blurred_depths > 0.1) & top_mask
s7_blurred_depths[valid_pixels] = blurred[valid_pixels]
blurred = cv2.GaussianBlur(s7_blurred_depths, (5, 5), 0)
valid_pixels = (s7_blurred_depths > 0.1) & top_mask
s7_blurred_depths[valid_pixels] = blurred[valid_pixels]
```

۴-۹ـ گام هشتم : برگرداندن مقادیر عمق به مقدار اصلی

در این قسمت ماتریس نقشه بدست آمده را از ۱۰۰ یا بیشترین فاصله در نظرگرفته شده ، کم می کنیم.

```
s8_inverted_depths = np.copy(s7_blurred_depths)
valid_pixels = np.where(s8_inverted_depths > 0.1)
s8_inverted_depths[valid_pixels] = \
    max_depth - s8_inverted_depths[valid_pixels]
depths_out = s8_inverted_depths
```

۴-۱۰ـ بیان نکات اضافه پیاده سازی

۴-۱۰-۱ـ تعریف کرنل ها

در فایل کرنل های استفاده شده در پیاده سازی با استفاده از کتابخانه numpy تعریف شده اند کد این کرنل ها را می توانید در فایل kernel.py ببینید.

۴-۱۰-۲ـ متریک ها

متریک های استفاده شده برای تست پیاده سازی دو متریک Mean Average Error و Root Mean Square Error می باشند کد این متریک ها در تابع زیر آورده شده است.

```
def calculate_metrics_mm(self, output, gt_item):

    valid_mask = gt_item > 0.1
    output_mm = 1e3 * output[valid_mask]
    gt_mm = 1e3 * gt_item[valid_mask]
    diff = np.abs(output_mm - gt_mm)
    mse = np.mean(np.power(diff, 2))
    rmse = np.sqrt(mse)
    mae = np.mean(diff)
    return rmse, mae
```

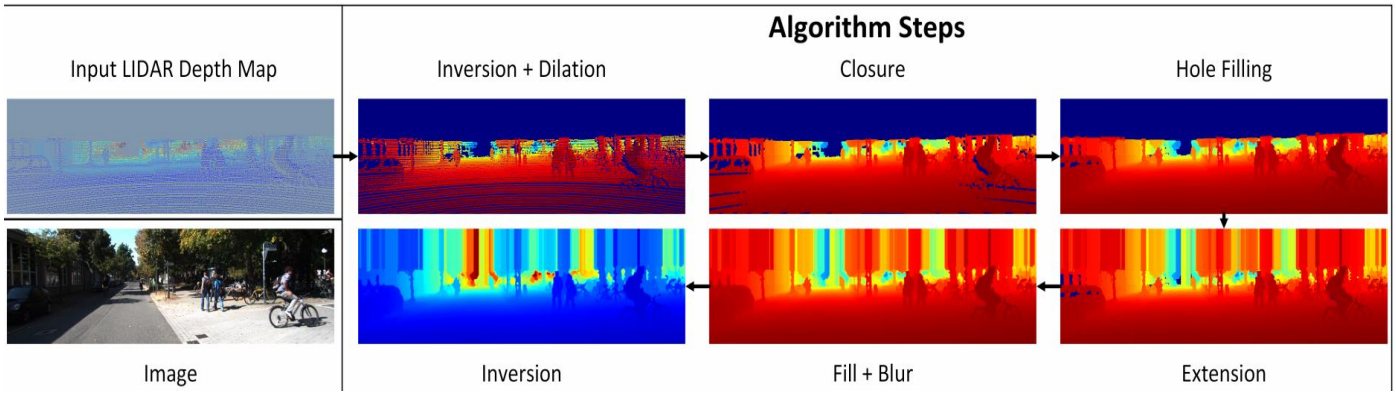
۴-۱۰-۳ـ رنگی کردن نقش ها

برای رنگی کردن نقشه های بدست آمده از تابع cv2.applyColorMap کتابخانه opencv استفاده شده است تا دید کامل تر و جذاب تری از نقشه های عمق بدست آمده ، ارائه دهیم.

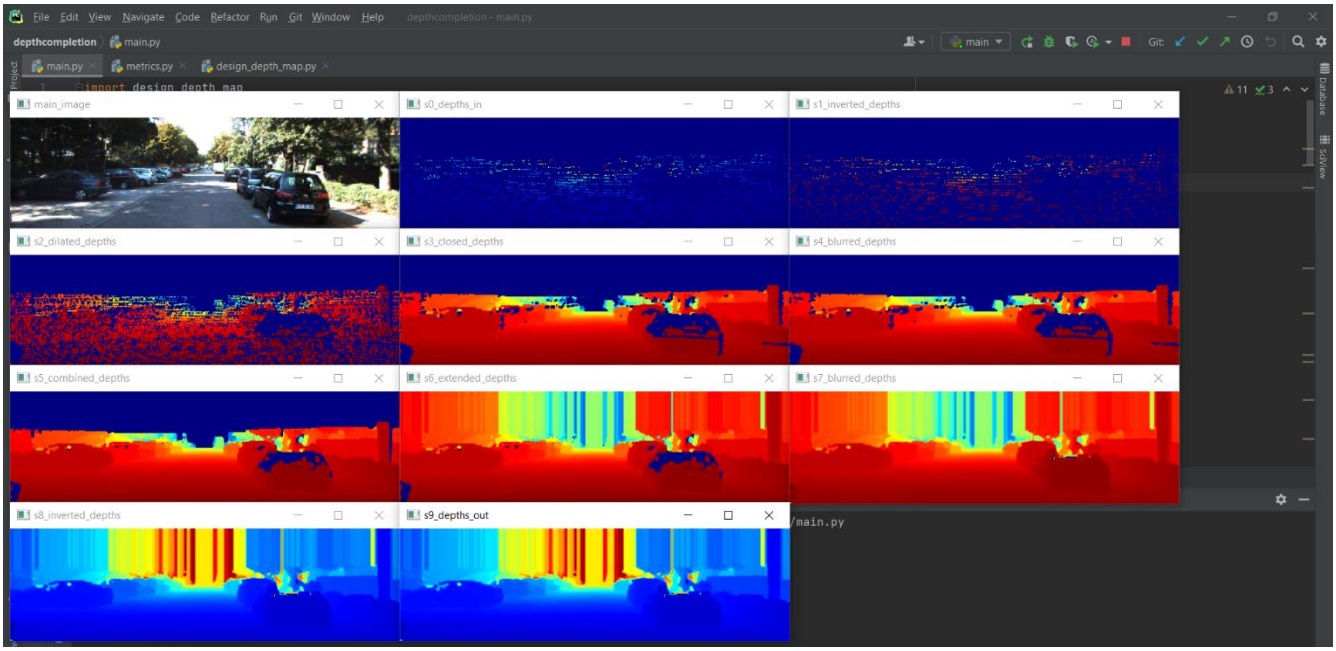
```
cv2.applyColorMap(
    np.uint8(value / np.amax(value) * 255),
    cv2.COLORMAP_JET)
```

۴-۱۱ـ مقایسه نتایج بصری پیاده سازی با نتایج مقاله

نتایج نهایی پیاده سازی در شکل زیر آورده شده است و از لحاظ بصری با نتایج پیاده سازی خود مقاله تفاوتی ندارد، شکل ۱۹ نتایج پیاده سازی نویسنده مقاله و شکل ۲۰ نتایج پیاده سازی نویسنده گزارش می باشد.



شکل ۱۹ – نتایج پیاده سازی مقاله



شکل ۲۰ – نتایج پیاده سازی نویسنده گزارش

فصل پنجم : بحث و نتیجه گیری

۵-۱ـ پیشگفتار

در این فصل ابتدا نتایج نویسندگان مقاله [1] مورد بررسی قرار می گیرد سپس نتایج پیاده سازی نویسنده گزارش با نتایج مقاله مورد مقایسه قرار خواهد گرفت و در نتها نتایج مقاله مورد گزارش با یکی از مطرح روش های حوزه تکمیل عمق یعنی PENet [9] مقایسه میشود.

۵-۲ـ شاخص های اندازه گیری

بنچمارک تکمیل عمق KITTI شاخص های فراوانی برای این نمونه مسئله در نظر گرفته است منتها این مقاله دو شاخص اصلی RMSE و MAE را در نظر می گیرد.

شاخص های RMSE و MAE بصورت زیر تعریف می شوند :

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum |d_i - d_i^*|^2}$$

$$\mathbf{MAE} = \frac{\mathbf{1}}{\boldsymbol{N}}|\boldsymbol{d_i} - \boldsymbol{d_i^*}|$$

۵-۳ـ بررسی نتایج مقاله

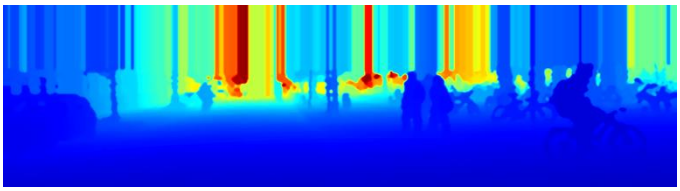
همانطور که در جدول زیر مشاهده می کنید روش ارائه شده در مقاله مورد گزارش در اوایل سال ۲۰۱۸ در سه شاخص اندازه گیری از چهار شاخص توانسته نمره برتری را بگیرد دو روش sparseCONVS و NN+CNN [11] که مبتنی بر الگوریتم های قدرتمند یادگیری عمیق هستند هم نتوانسته اند عملکردی در حد این مدل داشته باشند. در قسمت ۵ـ۶ نتایج این روش با یکی از مطرح ترین روش های حوزه تخمین عمق یعنی PENet مورد مقایسه قرار خواهد گرفت.

Method	iRMSE (1/km)	iMAE (1/km)	RMSE (mm)	MAE (mm)	Runtime (s)
NadarayaW	6.34	1.84	1852.60	416.77	0.05
SparseConvs	4.94	1.78	1601.33	481.27	**0.01**
NN+CNN	**3.25**	**1.29**	1419.75	416.14	0.02
Ours (IP-Basic)	3.78	**1.29**	**1288.46**	**302.60**	0.011

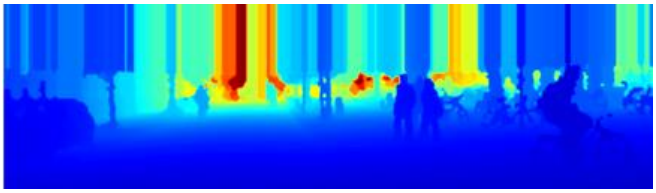
جدول شماره ۱ – مقایسه الگوریتم ارائه شده با بهترین مدل های زمان ارائه این روش ها روی دادگان تست بنچمارک KITTI آزمایش شده اند.

۵-۳ـ مقایسه نتایج پیاده سازی با نتایج مقاله

از لحاظ بصری نتیجه نهایی پیاده سازی نویسنده گزارش و نتایج مقاله [1] تقریبا مشابه هم هستند. در شکل ۲۱ می توانید نتایج را بصورت بصری مشاهده کنید.



الف



ب

شکل ۲۱ – الف) نتیجه پیاده سازی نویسنده گزارش – ب) نتیجه پیاده سازی مقاله

در جدول دو نتایج پیاده سازی و مقاله آورده شده است در این جدول نتایج پیاده سازی بر اساس دیتاست KITTI Validation Cropped می باشد ولی نتایج مقاله بر اساس دیتاست برروی مجموعه تست همین دیتاست اعمال شده است که برای تست پیاده سازی روی آن مجموعه دادگان Ground Truth آن در دسترس نبود. و احتمالا علت اختلاف نتایج نیز همین موضوع می باشد.

	Dataset	RMSE	MAE
Paper Implementation	KITTI Test Set	1288.46	302.6
My Implementation	KITTI Validation Cropped	1651.34	696.7

جدول ۲ – نتایج مقاله و نتایج پیاده سازی

۵-۴ـ مقایسه نتایج الگوریتم مقاله مورد گزارش با مقاله الگوریتم PENet

همانطور که در جدول شماره ۳ مشاهده می کنید الگوریتم مقاله مورد گزارش از لحاظ شاخص های RMSE و MAE هیچ حرفی برای گفتن ندارد و تقریبا این دو مدل بطورکلی قابل مقایسه نیستند حتی مدل کلاسیک از لحاظ سرعت نیز کندتر از مدل PENet می باشد.

	RMSE	MAE	Run Time
PENet[9]	730.08	210	0.04
Classic Method [1]	1288.46	302.6	0.011

جدول ۳ – مقایسه نتایج مدل PENet و مدل مقاله مورد گزارش

۵-۵ـ مشکلات الگوریتم مقاله [1]

اگر دیدگاه ها و وبسایت های مختلف را پیرامون این الگوریتم مورد بررسی قرار دهید تمامی آن ها به یک نکته در این الگوریتم اشاره می کنند. آن هم اینکه این الگوریتم فقط برای نقشه های تنک عمیق با توزیع UNIFORM همانند KITTI DEPTH COMPLETION مناسب و زمانیکه توزیع این نقشه ها تغییر می کند این الگوریتم فاقد کارایی خواهد بود و نتایج خوبی را به همراه نخواهد داشت.

جمع بندی

در این گزارش ابتدا یک مقدمه و پیشینه کامل از موضوع تخمین عمق و زیرشاخه آن یعنی تکمیل عمق ارائه شد سپس جزئیات الگوریتم ارائه شده توسط مقاله مورد گزارش بطور کامل مورد بررسی قرار گرفت همچنین یک فصل به فرآیند پیاده سازی این الگوریتم اختصاص پیدا کرد. بطورکلی الگوریتم مورد بحث یک روش سریع برای تکمیل عمق از نقشه های تنک عمق ارائه میدهد این روش در سال ۲۰۱۸ تقریبا جز بهترین ها بوده است اما به مرور و با پیشرفت یادگیری عمیق شبکه هایی ایجاد شدند که عملا از چنین الگوریتمی را در بسیاری از موارد استفاده نمی شود منتها این این الگوریتم همچنان جز سریع ترین روش های تکمیل عمق براساس نقشه های تنک عمیق می باشد. مزیت بسیار مهم دیگری که این الگوریتم نسبت به روش های یادگیری عمیق دارد موضوع عدم نیاز به دادگان آموزشی است. در نهایت الگوریتم ارائه شده برای کاربری های خاصی قابل استفاده می باشد.

منابع و ماخذ

[1] Jason Ku, Ali Harakeh, Steven L. Waslander, In Defense of Classical Image Processing: Fast Depth Completion on the CPU,2018, https://arxiv.org/abs/1802.00036v1.

[2]M.Mortazavi, H.Hassanpour, A.Pouyan, Depth Detection in Digital Images, Master Thesis

[3]Jae-Il J, Jong-Ho L, In-Yong S, Ji-Hee M, and Yo-Sung H. (2010), "Improved Depth Perception of Single-view", ECTI TRANSACTIONS ON ELECTRICAL ENG, ELECTRONICS, AND COMMUNICATIONS.

[4]Mirzabaki M. (2004) " Depth Detection Through Interpolation Functions", WSCG posters proceedings. WSCG_2004, P2-6, Plzen, Czech Republic.Copyright UNION Agency ' Science Press.

[5]Nedovi´c V, Smeulders A.W.M, Redert A, Geusebroek J.M. (2007) "Depth Information by Stage Classification", IEEE.

[6]Salih Y, Malik A.S, May Z. (2011) "Depth Estimation Using Monocular Cues from Single Image", IEEE.

[7]Aslantas V. (2007) "A depth estimation algorithm with a single image", Optical Society of America.

[8]Kuo T, Lo Y, Lai Y. (2011) "Depth Estimation from a Monocular Outdoor Image", IEEE International Conference on Consumer Electronics (ICCE).

[9]Huang, SC., Kothari, T., Banerjee, I. et al. PENet—a scalable deep-learning model for automated diagnosis of pulmonary embolism using volumetric CT imaging. npj Digit. Med. 3, 61 (2020).

[10] Godard, Clément & Aodha, Oisin & Firman, Michael & Gabriel, Jourdan. (2019). Digging Into Self-Supervised Monocular Depth Estimation. 10.1109/ICCV.2019.00393.

[11] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. arXiv preprint arXiv:1708.06500, 2017. 1, 2, 4.