
بسم الله الرحمن الرحيم

دانشکده مهندسی مکانیک و مکاترونیک

پایان‌نامه کارشناسی ارشد مهندسی مکاترونیک

مدل‌سازی سه‌بعدی محیط داخلی به وسیله سنسور کینکت

محمد جواد طهماسبی

استاد راهنما :

دکتر حسین خسروی

شهریور ۱۳۹۷

شماره: ۸/۴۷/۱۲۸
تاریخ: ۹۷/۷/۲

باسمه‌تعالی

مدیریت تحصیلات تکمیلی

فرم شماره (۳) صورتجلسه نهایی دفاع از پایان نامه دوره کارشناسی ارشد

با نام و یاد خداوند متعال، ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد آقای محمد جواد طهماسبی با شماره دانشجویی ۹۴۱۱۶۱۴ رشته مهندسی مکانیک گرایش مکاترونیک تحت عنوان مدل سازی سه‌بعدی محیط داخلی به وسیله سنسور کینکت که در تاریخ ۹۷/۶/۱۴ با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار گردید به شرح ذیل اعلام می گردد:

قبول (با درجه: خیلی خوب) ☑ مردود ☐

نوع تحقیق: نظری ☐ عملی ■

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضاء
۱ـ استاد راهنمای اول	دکتر حسین خسروی	استادیار	
۲- استاد راهنمای دوم	ـــــ	ـــــ	ـــــ
۳- استاد مشاور	ـــــ	ـــــ	ـــــ
۴- نماینده تحصیلات تکمیلی	دکتر مسعود مهدی زاده رخی	استادیار	
۵- استاد ممتحن اول	دکتر علیرضا احمدی فرد	دانشیار	
۶-ـ استاد ممتحن دوم	دکتر امیر حسین نایبی آستانه	استادیار	

نام و نام خانوادگی رئیس دانشکده: دکتر محمد محسن شاه مردان

تاریخ و امضاء و مهر دانشکده:

تبصره: در صورتی که کسی مردود شود حداکثر یکبار دیگر (در مدت مجاز تحصیل) می تواند از پایان نامه خود دفاع نماید (دفاع مجدد نباید زودتر از ۴ ماه برگزار شود).

تقدیم به پدرم

کوهی استوار و حامی من در طول تمام زندگی

و

تقدیم به مادرم

سنگ صبوری که الفبای زندگی به من آموخت

تقدیر و تشکر:

«من لم یشکر المخلوق لم یشکر الخالق»

منت خدای را عز و جل که طاعتش موجب قربت است و به شکر اندرش مزید نعمت. هر نفسی که فرو می‌رود ممد حیات است و چون برمی‌آید مفرح ذات. پس در هر نفسی دو نعمت موجود است و بر هر نعمتی شکری واجب.

با سلام و صلوات بر پیامبر اعظم حضرت محمد مصطفی(ص) و خاندان پاکش و با سپاس و حمد بی‌همتای خداوند متعال که انسان را خلق کرد و به‌وسیله قلم، تعلیم نمود و به انسان آنچه را که نمی‌دانست آموخت.

حال که به یاری و نصرت حضرت حق‌تعالی، این پایان‌نامه به اتمام رسیده است، بر خود واجب دانستم تا از تلاش صادقانه و خستگی‌ناپذیر استاد ارجمند، جناب آقای دکتر حسین خسروی که به‌عنوان استاد راهنما در جهت هدایت و راهبری این پایان‌نامه متحمل زحمات زیاد شدند و با پیگیری‌های مسئولانه خویش موجب تکمیل این پایان‌نامه گردیدند، کمال تشکر و قدردانی را دارم.

درنهایت مراتب تشکر صمیمانه خود را به اعضاء خانواده‌ام که در طول دوران تحصیل و در مدت تهیه پایان‌نامه مشکلات و سختی‌های گوناگونی را با آغوش باز پذیرا شدند و محیطی آرام را رقم زدند، ابراز می‌کنم.

این قلم وام‌دار خیل این عزیزان بزرگوار است و از درگاه خداوند سبحان برای تمامی این عزیزان آرزوی موفقیت و سلامتی روزافزون خواستارم.

محمد جواد طهماسبی

تعهــد نامه

اینجانب **محمد جواد طهماسبی** دانشجوی دوره کارشناسی ارشد رشته مهندسی مکاترونیک دانشکده مهندسی مکانیک و مکاترونیک دانشگاه صنعتی شاهرود نویسنده پایان‌نامه «**مدل‌سازی سه‌بعدی محیط داخلی به وسیله سنسور کینکت**» تحت راهنمایی دکتر حسین خسروی متعهد می‌شوم:

- تحقیقات در این پایان‌نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های محققان دیگر به مرجع مورد استفاده استناد شده است.
- مطالب مندرج در پایان‌نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است.
- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی شاهرود می‌باشد و مقالات مستخرج با نام «دانشگاه صنعتی شاهرود» و یا «Shahrood University of Technology» به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان‌نامه تأثیرگذار بوده‌اند در مقالات مستخرج از پایان‌نامه رعایت می‌گردد.
- در کلیه مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده (یا چینی‌جاهای آن‌ها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل رازداری، ضوابط و اصول اخلاق انسانی رعایت شده است.

تاریخ:

امضای دانشجو :

چکیده:

در این پایان‌نامه، یک سیستم باز سازی سه‌بعدی مبتنی بر داده‌های RGB-D را توسعه داده‌ایم. سعی ما بر این بوده است که راه‌کار پیشنهادی هم برای مدل‌سازی سه‌بعدی اشیا و هم برای بازسازی محیط داخلی مناسب باشد.

ابتدا داده‌های RGB-D را به وسیله سنسور کینکت که با دست نگاه داشته شده است ضبط می‌کنیم. عمق بدست آمده از سنسور در مرحله پیش پردازش پالایش شده و نویز آن تا حدی حذف می شود. پارامترهای خارجی دوربین در حرکت بین دو قاب (شامل ماتریس چرخش و بردار جابجایی) را بدست آورده و بعد از آن ردیابی مسیر حرکت دوربین را، با استفاده از یک الگوریتم هم‌ترازی سه‌بعدی مقاوم مبتنی بر RANSAC، انجام می دهیم. سپس میزان حرکت دوربین را به منظور افزایش دقت نقشه‌برداری بررسی می‌کنیم. نقشه ابتدایی محیط، که شامل حالت‌های دوربین بین دو قاب، ویژگی‌ها و مشاهدات از صحنه است، با استفاده از ردیابی قاب به قاب ساخته شده و به نقشه کلی اضافه می‌شود.

به منظور کاهش میزان مصرف حافظه و بهبود زمان پردازش، به جای آنکه تمام اطلاعات پیکسل‌های ابر نقطه سه‌بعدی را ذخیره کنیم، از واکسل‌ها استفاده کرده و حجم ابر نقطه را کاهش می‌دهیم.

در نهایت با توجه به اینکه سنسور کینکت برای هر پیکسل مقدار عمق دقیق را می‌دهد، نقشه سه‌بعدی تولید شده محیط داخلی در مقیاس واقعی است، که می‌تواند برای بازرسی‌های بصری از راه دور، فرآیندهای اندازه‌گیری و مهندسی معکوس مفید باشد.

کلمات کلیدی: بازسازی سه بعدی، مدل سازی سه بعدی، کینکت

فهرست مطالب

فهرست شکل‌ها

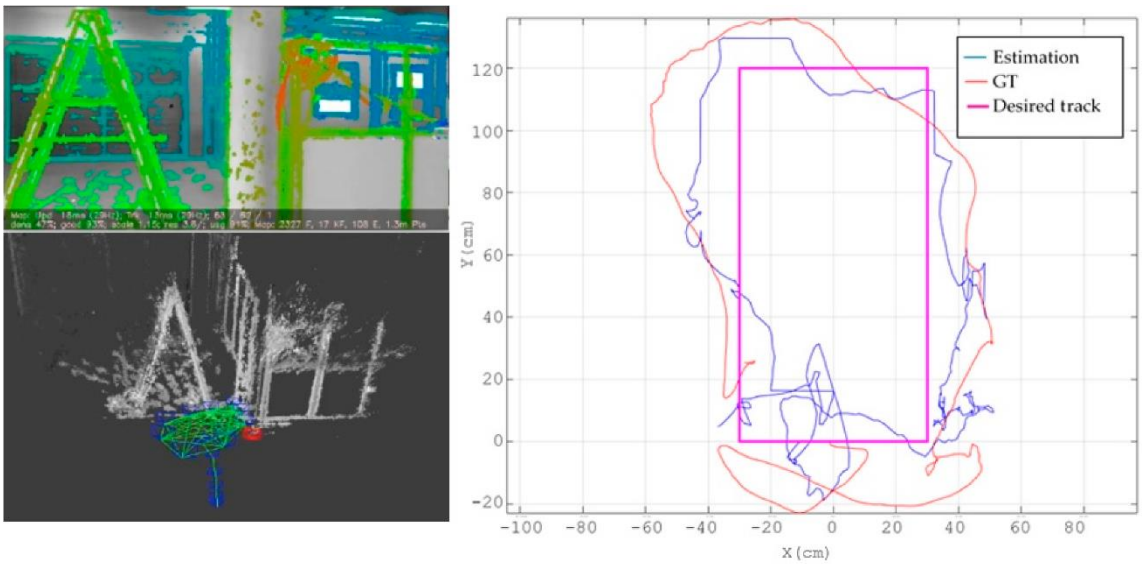
فهرست جدول‌ها

فهرست الگوریتم‌ها

فصل اول: مقدمه

۱-۱ پیشگفتار

باز سازی دیجیتال مدل‌های سه‌بعدی مبتنی بر دوربین، با ا ستفاده از ت صاویر محیط و اج سام در دنیای واقعی، به یک مسئله چالشی در طول دهه‌های گذشته در حوزه بینایی ماشین تبدیل شده است. هدف ا صلی این ا ست که مدل سه بعدی یک شی یا محیط را با ترکیب کردن اطلاعات مختلف از نماهای متفاوت دوربین و اضافه کردن آن‌ها به یک نقشه سراسری بسازیم. بنابراین مسیر دوربین و مدل سه بعدی باید به طور هم‌زمان تخمین زده شوند، که به این مسئله SLAM[1] گفته می‌شود. SLAM الگوریتم اصلی برای بازسازی سه بعدی محیط‌های دنیای واقعی است. (شکل۱-۱)

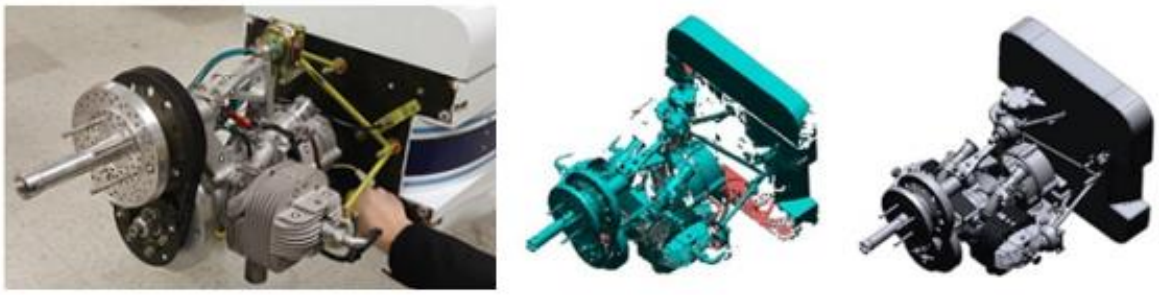


شکل(۱-۱) (سمت چپ) بازسازی محیط نقشه‌برداری شده(سمت راست) مسیر تخمینی که دوربین طی کرده است[۱]

یکی از زمینه‌های کاربردی برای بازسازی سه بعدی، حوزه رباتیک است. زمانی که یک ربات سیار[2] می‌خواهد یک محیط ناشناخته را فقط بر اساس اطلاعات دوربین نصب شده بر روی آن پیمایش کند، نیاز دارد تا مدل سه بعدی محیط را بسازد. این موضوع، به این خاطر مهم است که ربات باید محل خود را تخمین بزند و از برخورد با اشیا در محیط پیش رویش جلوگیری کند. علاوه بر این تولید مدل‌های سه بعدی برای کاربردهایی مانند بازی‌سازی یا مقاصد پزشکی مفید هستند.

[1] Simultaneous localization and mapping
[2] Mobile robot

در کنار سناریوهای گفته شده در بالا مهندسی معکوس یکی از مهم‌ترین موارد استفاده بازسازی سه بعدی در صنعت است. به طور کلی، مهندسان معمولا به مدل‌های سه بعدی اشیاء واقعی برای شبیه‌سازی و اهداف اندازه‌گیری یا حتی تولید دوباره‌ی اشیاء واقعی نیاز دارند. این مدل‌های شبیه‌سازی شده باید اطلاعات دقیق هندسی درباره اشیا واقعی را فراهم کنند ولی در عین حال ساختن این مدل‌ها با استفاده از نرم‌افزارهای رایج CAD[1] پیچیده و زمان‌بر است. بنابراین برای تولید خودکار مدل CAD، روش‌های بازسازی سه‌بعدی زمان زیادی را ذخیره کرده و حتی ممکن است دقت بالاتری را نسبت به مدل طراحی شده دستی فراهم کند. شکل ۱-۲ نتیجه فرآیند مهندسی معکوس یک موتور را نشان می‌دهد [۲]



شکل (۱-۲) (سمت چپ) مهندسی معکوس یک موتور واقعی(وسط) به وسیله اسکن داده‌ها(سمت راست) مدل CAD ساخته شده

سنسور ارزان قیمت کینکت، که در دهه اخیر معرفی شده، باعث افزایش تحقیقات در زمینه بازسازی سه بعدی شده است، به این خاطر که این سنسور تصویر رنگی و مقادیر عمق را به ازای هر پیکسل در نرخ قاب آنی[2] فراهم می‌کند. ابر نقاط[3] سه بعدی رنگی دقیق از محیط، که مستقیما توسط

[1] Computer aided design
[2] Real-time frame rate
[3] Points cloud

سنسور فراهم شده است، با راه‌کار SLAM ترکیب شده و فرآیند تولید مدل‌های واقعی[1] سه بعدی یک محیط را تسریع می‌کند.

به منظور ساخت یک ابر نقطه سه بعدی چگال[2] یک شی با دقت بالا، که مخصوصا برای بازسازی سه بعدی مهم است، اطلاعات از نماهای مختلف دوربین را نیاز داریم. این کار می‌تواند به وسیله حرکت دادن یک دوربین حول شی مورد نظر و اسکن کامل از تمام نماهای متفاوت صورت گیرد. ردیابی دوربین[3]، حرکت نسبی بین موقعیت فعلی و موقعیت قبلی را تخمین می‌زند و اجازه می‌دهد تا داده‌ها را از نماهای متفاوت به یک مدل سراسری اضافه کنیم. فرآیند بازسازی و ردیابی دوربین با هم ترکیب شده و در نهایت یک مدل سه‌بعدی دقیق را برای ما فراهم می‌کند.

۱-۲ انگیزه

در میان تعداد زیادی از کاربردهای بازسازی سه بعدی برای مهندسی معکوس، در این پایان‌نامه به دنبال بازسازی محیط داخلی هستیم که کاربردهای بسیاری دارد. اولا، تولید یک ابر نقطه از محیط داخلی به ما اجازه بازرسی آن محیط را بدون نیاز به اینکه خودمان در آن محیط باشیم، می‌دهد. ثانیا، مدل باید تا حد امکان چگال با شد تا بتواند جزئیات را به خوبی نشان دهد شکل۱-۳ مدل باز سازی شده یک اتاق را نشان می‌دهد که به وسیله روش بازسازی سه‌بعدی ساخته شده است.

[1] Metric models
[2] Dense
[3] Camera tracking

شکل (۱-۳) مدل سه بعدی محیط اتاق، که به وسیله اسکن سه‌بعدی ساخته شده است. این مدل اجازه بازرسی در ابعاد واقعی را می‌دهد

بازسازی دقیق این امکان را فراهم می‌کند تا ابعاد مدل دیجیتال ساخته شده را اندازه‌گیری کنیم، که این کار می‌تواند ساده‌تر از اندازه‌گیری اشیا واقعی توسط شخص باشد. در این پایان‌نامه، فرض می‌کنیم صحنه‌هایی که ساخته می شوند همیشه ثابت هستند و حرکت و تغییر شکلی در آن‌ها رخ نمی‌دهد. علاوه بر این، برای تسهیل بازسازی تغییری در محیط نمی‌دهیم. به این معنی که هیچ علامتی(مانند صفحه شطرنجی) در محیط قرار داده نشده و همچنین هیچ وسیله اضافی(مانند شتاب سنج) وجود ندارد تا به فرآیند بازسازی کمک کند.

برای تولید مدل سه بعدی محیط داخلی، سیستم بازسازی سه بعدی ما موقعیت نسبی دوربین را برای هر دو جفت قاب بدست آمده از سنسور RGB-D محاسبه می‌کند، که در نتیجه به ما اجازه می‌دهد داده‌های قاب‌ها را با هم یکی کرده و یک مدل سه بعدی سراسری را بسازیم. در این صورت مدل ما می‌تواند از تعداد زیادی قاب‌های کوچک ساخته شود. با این وجود موقعیت دوربین[1] در معرض خطا قرار دارد چرا که اندازه‌گیری‌های انجام شده دارای نویز است. این خطاهای کوچک، قاب به قاب با هم جمع می‌شوند. جمع این خطاهای جمع شونده می‌تواند باعث کاهش دقت در تخمین موقعیت‌های دوربین و در نتیجه، کاهش کیفیت مدل سه بعدی بازسازی شده بشود.

[1] Camera pose

۱-۳ بیان مسئله

هدف این پایان نامه توسعه یک نرم‌افزار RGB-D برمبنای بازسازی سه بعدی است که قادر باشد بازسازی دقیق مدل‌های سه بعدی چگال از محیط داخلی را فراهم کند.

در کنار آن یک سیستم SLAM ماژولار و منعطف بر مبنای داده‌های RGB-D توسعه داده شده است. در حالی که سیستم ارائه شده برای بازسازی محیط داخلی مناسب است ولی می‌تواند برای دیگر بازسازی‌های سه بعدی و مجموعه داده‌های مختلف استفاده شود.

۱-۴ ساختار پایان‌نامه

در این فصل به صورت خلاصه به هدف اصلی فرآیند بازسازی سه‌بعدی پرداختیم و مثال‌هایی از آن را نقل کردیم. ضرورت استفاده از بازسازی سه‌بعدی در مهندسی معکوس بیان شد. همچنین مسئله SLAM را بیان نمودیم و کاربردهایی از آن را ذکر کردیم.

در فصل دوم مروری بر کارهای پیشین صورت گرفته و روش‌های مختلف بازسازی سه‌بعدی ذکر گردیده است. فصل سوم به بیان مبانی نظری و ریاضی می‌پردازد. در فصل چهارم الگوریتم خود را شرح داده و به طور کامل مراحل آن را توضیح می‌دهیم. در فصل پنجم نتایج بدست آمده از این روش مورد بررسی و ارزیابی قرار می‌گیرد. و در نهایت در فصل ششم به اهداف آینده و روش‌هایی که می‌تواند کارایی الگوریتم را بهبود ببخشد می‌پردازیم.

فصل دوم: مروری بر کارهای پیشین

۲-۱ مقدمه

باز سازی سه بعدی ا شیا و محیط با این هدف که مدل ساخته شده به اندازه کافی چگال بوده و دارای جزئیات کافی باشد از طریق فرآیند SLAM بدست می‌آید. از آن جهت که فرآیندهای SLAM باید به طور همزمان مسیر دوربین و یک نقشه از محیط را بر اساس داده‌های نویزی تخمین بزنند، عدم دقت در نقشه و حالت‌های دوربین می‌تواند اتفاق بیافتد. بنابراین برای غلبه به این شرایط روش‌های متفاوتی در سالیان اخیر توسعه داده شده است. در ادامه آخرین روش‌های RGB-D SLAM و الگوریتم‌های بازسازی سه بعدی را می‌بینیم.

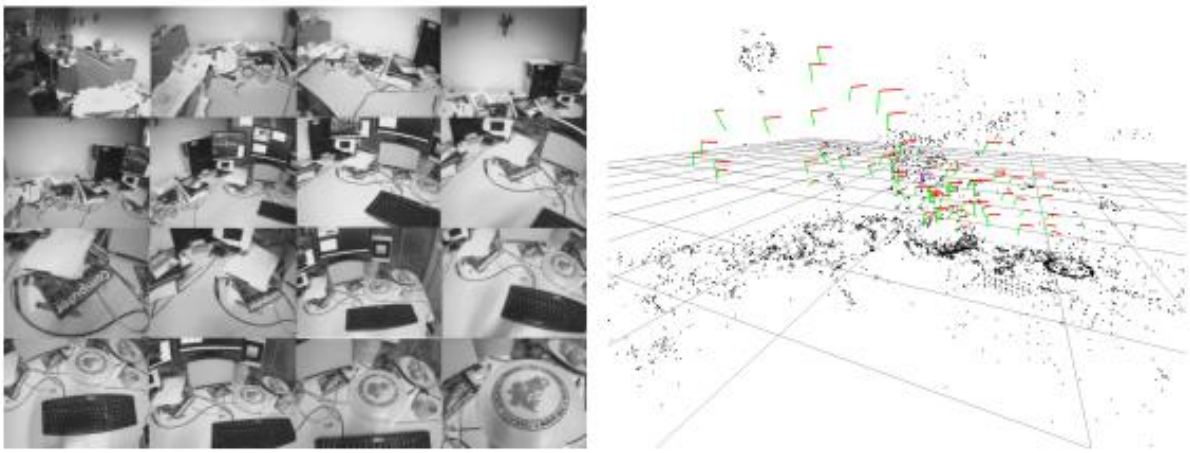
۲-۲ SLAM و بازسازی سه‌بعدی

مسئله SLAM تاریخچه طولانی در حوزه رباتیک دارد و از آن موقع تا کنون به خوبی بررسی شده است. در قلب این مسئله همانطور که قبلا گفته شد با استفاده از داده‌های بدست آمده از سنسور موقعیت دوربین به طور همزمان با نقشه محیطی که در آن حرکت می‌کنیم تخمین زده می‌شود.

اکثر سیستم‌های نقشه‌برداری سه بعدی یک روندنما[1] یکسان دارند. بعد از بدست آوردن تصویرهای رنگی و اطلاعات سه بعدی یا ترکیبی از هر دو به عنوان ورودی، ویژگی‌های پراکنده از تصویرهای رنگی استخراج شده و برای هم‌تراز کردن سه بعدی نسبی فضایی[2] بین جفت قاب‌ها استفاده می‌شوند. همچنین هم‌تراز کردن داده‌های عمق بدست آمده می‌تواند توسط الگوریتم ICP انجام شود. بعد از آن، خاتمه‌های حلقه[3] مسیر دوربین باید شناسایی شود. یعنی اگر تصویری را که قبلا دیده‌ایم دوباره ببینیم خاتمه حلقه باید لحاظ شود. در نهایت داده‌های قاب‌ها باید با توجه به یک سیستم مختصات کلی که نمایش دهنده مدل سه‌بعدی است به طور مقاوم[4] برهم منطبق شوند.

[1] Algorithm
[2] Spatial relative 3D alignment
[3] Loop closures
[4] Robust

در طول سال‌ها، راه‌کارهای نقشه برداری سه بعدی زیادی فقط با استفاده از داده‌های یک دوربین RGB توسعه داده شده است. این سناریوهای SLAM که فقط با استفاده از تصاویر رنگی باز سازی را انجام می‌دهند تحت عنوان SFM[1] شناخته می‌شوند[۳] و در روش‌هایی مانند MonoSLAM[4] یا PTAM[5]بیان شده است. این روش ردیابی و نقشه‌برداری دوربین را به طور موثر و به صورت آنی اجرا می‌کند، ولی نقشه‌های محیط بدست آمده توسط این روش به صورت پراکنده[2] است(شکل ۲-۱) .



شکل (۲-۱): الگوریتم SLAM تک دوربین که معمولا نقشه‌های پراکنده تولید می‌کند. (سمت چپ) مجموعه تصویرهای ورودی یک میز (سمت راست) نقشه تولید شده با نقاط ویژگی و حالت‌های دوربین[۵].

این نقشه‌ها فقط می‌توانند به صورت نسبی[3] باشند و اجازه باز سازی در واحد واقعی را نمی‌دهند. علاوه بر این در حالی که این روش‌ها برای ردیابی دوربین، دقیق هستند، ولی برای بدست آوردن مدل‌های سه بعدی چگال محدودیت دارند. برای غلبه بر این محدودیت، روش‌های جدید، به جای نقاط ویژگی پراکنده، هر پیکسل تصویر ورودی را برای ردیابی دوربین استفاده می‌کنند و سعی می‌کنند نقشه‌های چگال تولید کنند[۶, ۷].

[1] Structure from motion
[2] Sparse
[3] up to scale

از سنسورهای RGB-D ارزان قیمت، به طور خاص کینکت مایکروسافت[1]، برای حل بعضی چالش‌های SLAM تک دوربین، مانند اندازه‌گیری واقعی[2] استفاده می‌شود. دوربین‌های RGB-D تصویرهای RGB را به همراه اطلاعات عمق در نرخ قاب آنی تهیه می‌کنند. راه‌کارهای RGB-D SLAM توسعه داده شده، اطلاعات بصری از تصویرهای رنگی را با اطلاعات سه بعدی اندازه‌گیری شده از عمق برای نمایش محیط سه بعدی چگال ترکیب می‌کنند. مادامی که سنسورهای RGB-D عمق را در واحدهای واقعی و متریک فراهم می‌کنند این امکان هست که مدل سه بعدی اشیاء را مستقیما به صورت واقعی بسازیم و به این دلیل است که بر روی راه‌کارهای RGB-D تمرکز می‌کنیم.

۲-۲-۱ روش Kinect fusion

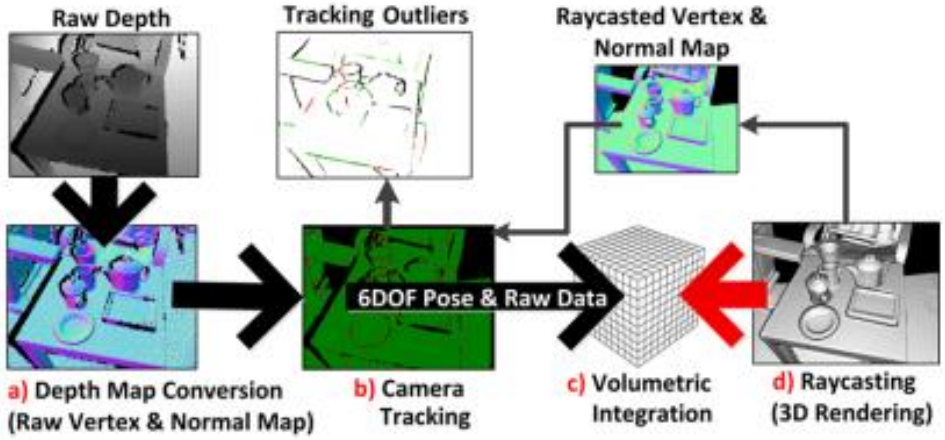
سیستم Kinect fusion در سال ۲۰۱۱ به وسیله NewCombe توسعه داده شده است[۶]. این روش از یک دوربین کینکت برای ساخت صحنه‌های پیچیده در مقیاس محیط داخلی به صورت آنی استفاده می‌کند. کارایی بالای سیستم به وسیله استفاده گسترده از روش‌های برنامه‌نویسی GPGPU[3] بدست می‌آید. این روش فقط از اطلاعات عمق استفاده کرده و نیازی به تصویر رنگی ندارد. در قلب این روش، مدل ساخته شده در یک نمایش حجمی[4] سراسری به فرم معکب‌های سه بعدی که واکسل نامیده می‌شوند ذخیره می‌گردد. تمام عمق‌های اندازه‌گیری شده که به وسیله سنسور از صحنه مشاهده شده، بدست آمد به یک مدل سطح ضمنی که به حجم TSDF[5] شناخته می‌شود به صورت آنی ترکیب می‌گردد. شکل ۲-۲ مدل سه بعدی ساخته شده توسط کینکت فیوژن را نشان می‌دهد.

ردیابی دوربین به وسیله تخمین موقعیت فعلی دوربین با توجه به مدل چگال TSDF در نرخ قاب ۳۰ هرتز انجام می شود. بنابراین، قاب فعلی دوربین به یک قاب از مدل منطبق می شود، که به صورت

[1] Microsoft kinect
[2] Metric measurement
[3] General Purpose Graphics Processing Unit
[4] Global volumetric
[5] Truncated Signed Distance Function

ترکیبی به وسیله TSDF ray-casting ، با استفاده از یک الگوریتم ICP[1] سریع در راه‌کار کل به جزء[2] تولید شده است. با استفاده از این موقعیت دوربین تخمین زده شده، قاب فعلی به TSDF اضافه شده، به طوری که مدل سطح سه‌بعدی به طور پیو سته به روزر سانی شده و به طور کامل ترکیب می شود. معماری کینکت فیوژن در شکل ۲-۳ نمایش داده شده است.

شکل (۲-۲): (الف) تصویر RGB، (ب) نرمال‌های سطح و (پ) سطح بازسازی شده نویزدار. (ت) و (ث) نرمال‌های سطح و مدل Phong Shaded مدل سه‌بعدی کینکت فیوژن را نشان می‌دهد[۶].

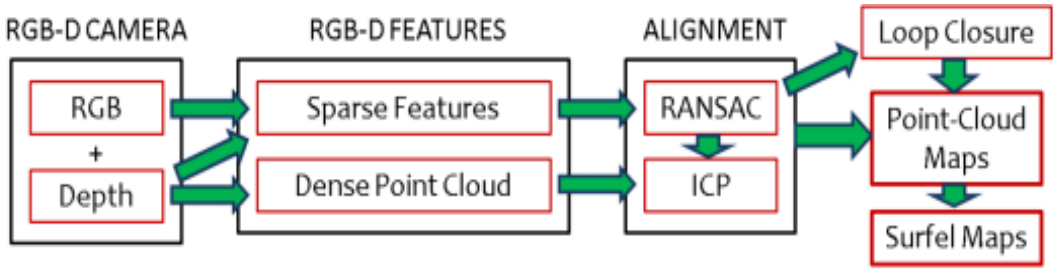


شکل (۲-۳): مراحل اصلی ردیابی کینکت فیوژن و خط لوله بازسازی[۶].

[1] Iterative closet point
[2] Coarse-to-fine

۲-۲-۲ نقشه‌برداری RGB-D

در حالی که کینکت فیوژن هدفش ساخت مدل‌هایی در ابعاد یک اتاق به صورت آنی است، راه‌کار نقشه‌برداری Henrry [۸] بر روی نقشه‌برداری محیط داخلی وسیع متمرکز است. سراسر ساختار این روش طوری طراحی شده است تا SLAM را با سنسورهای RGB-D پیاده‌سازی کند(شکل ۲-۴).



شکل (۲-۴): الگوریتم نقشه‌برداری RGB-D که ویژگی‌های پراکنده و چگال ICP مبتنی بر هم‌ترازی سه‌بعدی را برای تخمین نسبی حالت فراهم می‌کند[۸].

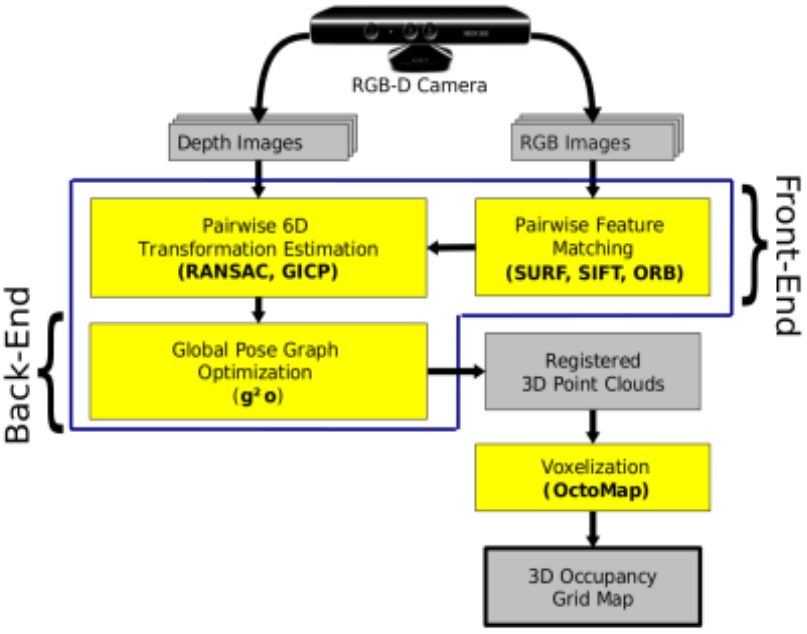
برای ردیابی دوربین، یک روش ترکیبی که RGB-D ICP نامیده می‌شود برای بدست آوردن هم‌ترازی نسبی بین قاب‌های متوالی تو سعه داده شده است. این راه‌کار ابتدا یک هم‌ترازی نسبی بر مبنای تناظرهای نقاط ویژگی[1] پراکنده تصویر رنگی اجرا می‌کند(شکل ۲-۵). اولین هم‌ترازی مبتنی بر ویژگی بدست آمده به عنوان مقدار اولیه تخمین حالت به الگورینم ICP داده می‌شود که به اطلاعات عمق بستگی دارد. خاتمه حلقه هم بر مبنای تطبیق ویژگی‌های قاب فعلی با بعضی از قاب‌های قبلی اجرا می شود که از نظر فضایی نزدیک هستند. در مواردی که هم‌ترازی موفق باشد، حالت نسبی بین قاب‌های متناظر را تخمین زده و به گراف حالت[2] اضافه می‌کنیم. برای بهبود کلی نقشه ساخته شده یک بهینه سازی گراف حالت نهایی اجرا می شود. نقشه برداری RGB-D برای کاربردهای آنی مناسب نیست چون از GPU استفاده نمی‌کند.

[1] Correspondences feature point
[2] Pose graph

شکل (۲-۵): (سمت چپ) نقاط ویژگی پراکنده بدست آمده به وسیله روش FAST (سمت راست) همان ویژگی‌ها در حالت سه‌بعدی نمایش داده شده‌اند[۸].

۲-۲-۳ روش RGB-D SLAM

سیستم RGB-D SLAM که توسط Endres و همکارانش در [۹, ۱۰] ارائه شده است خیلی شبیه به روش نقشه‌برداری RGB-D است که در بخش ۲-۲-۲ درباره آن صحبت کردیم. با این وجود راه‌کار اساسی این روش که در شکل نشان داده شده است در ردیابی دوربین و ارائه مدل انتخاب شده متفاوت است(شکل ۲-۶).



شکل (۲-۶): نمای شماتیک سیستم RGB-D SLAM. قسمت SLAM front-end حالت‌های نسبی بین قاب‌ها را با استفاده از هم‌ترازی سه‌بعدی مبتنی بر ویژگی تخمین می‌زند در حالی که SLAM back-end بهینه‌سازی حالت گراف را اجرا می‌کند[۹].

محلی سازی دوربین مانند روش نقشه‌برداری RGB-D محاسبه می شود ولی به صورت ساده‌تر شده. برای هر ورودی تصویر RGB بدست آمده از سنسور RGB-D، ویژگی‌های بصری استخراج شده و با ویژگی‌های تصویر رنگی قاب قبلی تطبیق داده می‌شوند. با استفاده از اطلاعات عمق و ویژگی‌های تطبیق یافته، تناظرهای سه بعدی point-wise بین قاب‌ها می‌تواند بنا شود. این تناظرهای سه بعدی اجازه می‌دهد تا به طور مقاوم[1] هم‌ترازی[2] سه بعدی نسبی بین قاب‌ها را با استفاده از RANSAC[3] تخمین بزنیم. خاتمه‌های حلقه همانند آن چیزی که در روش نقشه‌برداری RGB-D گفته شد شناسایی می‌شوند. از آنجا که ردیابی قاب به قاب نتیجه‌اش خطای جمع شونده[4] در حالت‌های تخمین زده شده دوربین است، سازگاری سراسری[5] به وسیله بهینه‌سازی توابع خطای غیرخطی مبنی بر گراف بدست می‌آید [۱۱].

[1] Robust
[2] Alignment
[3] Random sample consensus
[4] Accumulated error
[5] Global consistency

فصل سوم: مباحث نظری

۳-۱ مقدمه

این فصل مباحث ریاضی مورد نیاز را که در فصل‌های بعدی پایان نامه لازم است، بر اساس[۱۲, ۱۳] فراهم می‌کند. همچنین نحوه بدست آوردن پارامترهای داخلی دوربین را با استفاده از روش کالیبراسیون توضیح می‌دهیم. در انتهای این فصل نیز به معرفی سنسور کینکت می‌پردازیم. با این وجود چون بعضی از مباحث را نمی‌توان به صورت کامل بیان کرد ما خواننده را برای مطالعه بیشتر به منابع مربوطه ارجاع می‌دهیم.

۳-۲ نقاط و تبدیلات

نقاط به عنوان مهم‌ترین نوع داده‌ی هندسی شناخته شده و به وسیله بردار نمایش داده می‌شوند. یک نقطه $\boldsymbol{x}$ در فضای دوبعدی به وسیله بردار $\boldsymbol{x} = (x.y)^T \in \mathbb{R}^2$ یا در مختصات همگن به وسیله بردار $\bar{\boldsymbol{x}} = (x.y.1)^T$ نشان داده می شود. به طور مشابه یک نقطه سه بعدی $\boldsymbol{X}$ می‌تواند به صورت $\boldsymbol{X} = (X.Y.Z)^T \in \mathbb{R}^3$ یا در مختصات همگن به وسیله بردار چهارتایی $\bar{\boldsymbol{X}} = (X.Y.Z.1)^T$ نشان داده شود. یک تبدیل اقلیدسی سه بعدی[1] شامل چرخش و انتقال در فضای سه بعدی بوده و بنابراین دارای شش درجه آزادی[2] است. به این خاطر که این تبدیل اندازه و جهت[3] هر جفت نقطه یک جسم صلب را حفظ می‌کند از اینرو حرکت جسم صلب[4] نیز شناخته می شود. مجموعه حرکت‌های جسم صلب یک گروه Lie را شکل می‌دهند که به گروه اقلیدسی خاص[5] معروف بوده و با عبارت $SE(3)$ نشان داده شده و به طور کامل در [۱۴] توضیح داده شده است.

$$SE(3) = \left\{T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \in R^{4\times4} | R \in SO(3). t \in R^3\right\} \qquad ۱-۳$$

در حالی که قسمت انتقالی حرکت جسم صلب $g = (R.t) \in SE(3)$ به وسیله بردار انتقال $t \in \mathbb{R}^3$ تعریف می شود، نمایش‌های متفاوتی برای ارائه قسمت چرخش وجود دارد $R \in SO(3)$،

[1] 3D Euclidean transformation
[2] 6 Degree of Freedom
[3] Orientation
[4] Rigid body motion
[5] Special Euclidean group

که $SO(3)$ گروه ارتوگونال ویژه[1] نمایش دهنده چرخش اســـت. در این پایان‌نامه از ماتریس ۳ × ۳ چرخش استفاده می‌کنیم.

یک ماتریس چرخش ارتوگونال 3×3 $R \in \mathbb{R}^{3x3}$ (با $|R| = 1$ و $RR^T = I$) شـامل ۹ پارامتر و دارای ســـه درجه آزادی اســـت. هر دو ماتریس t و R می‌توانند در یک ماتریس تبدیل ۴ × ۴ با هم ترکیب شده و حرکت جسم صلب که با عبارت $g \in SE(3)$ مشخص می‌شود را توصیف کنند[۱۴]:

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ۲\text{-}۳$$

علاوه بر این ما تبدیل[2] $\mathcal{T}$ که نقطه ســـه بعدی $P = (X.Y.Z)^T \in R^3$ را به $\acute{P} = (\acute{X}.\acute{Y}.\acute{Z})^T$ تبدیل می‌کند به صورت زیر تعریف می‌کنیم[۱۴]:

$$\begin{aligned} \mathcal{T}: SE(3) \times \mathbb{R}^3 \longrightarrow \mathbb{R}^3 . P' = \mathcal{T}(g.\boldsymbol{P}) &= RP + t \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_Z \end{pmatrix} \end{aligned} \qquad ۳\text{-}۳$$

تبدیل معکوس $\mathcal{T}^{-1}$ نقطه P' را به P بر میگرداند

$$\begin{aligned} \mathcal{T}^{-1}: SE(3) \times \mathbb{R}^3 \longrightarrow \mathbb{R}^3 . \boldsymbol{P} &= \mathcal{T}^{-1}(g.\boldsymbol{P'}) \\ &= R^T(\boldsymbol{P'} - t) \end{aligned} \qquad ۴\text{-}۳$$

نقطه ســـه بعدی همگن $\bar{P} = (X.Y.Z.1)^T \in \mathbb{R}^4$ را می‌توان با ضـــرب کردن در یک ماتریس تبدیل 4×4 به نقطه $\bar{P}' = (X'.Y'.Z'.1)^T$ تبدیل کرد:

$$\mathcal{T}': SE(3) \times \mathbb{R}^4 \longrightarrow \mathbb{R}^4 . \bar{\boldsymbol{P}}' = \mathcal{T}'(g.\bar{\boldsymbol{P}}) = T\boldsymbol{P} \qquad ۵\text{-}۳$$

[1] Special orthogonal group
[2] Transformation

با توجه به اینکه $R^{-1} = R^T$ اســت، تبدیل معکوس $\mathcal{T}^{-1}$ ماتریس 4×4 به صــورت زیر نوشــته می‌شود

$$T^{-1} = \begin{bmatrix} R^T & -R^T t \\ 0 & 1 \end{bmatrix} \quad ۶\text{-}۳$$

و البته می‌توان برای ساخت تبدیل معکوس نقاط سه بعدی همگن طبق رابطه زیر عمل کرد:

$$\mathcal{T}'^{-1}: SE(3) \times \mathbb{R}^4 \longrightarrow \mathbb{R}^4 . \overline{\boldsymbol{P}} = \mathcal{T}^{-1}(g . \overline{\boldsymbol{P}}') = T^{-1}\overline{\boldsymbol{P}}' \quad ۷\text{-}۳$$

مزیت اســـتفاده از ماتریس‌های همگن به شـــکل $T \in \mathbb{R}^{4\times4}$ این اســـت که تبدیل‌های متوالی می‌توانند به سادگی در ماتریس تبدیل مربوطه ضرب شده و در یک ماتریس تبدیل واحد قرار گیرند. این مزیت به ما این امکان را می‌دهد تا یک تبدیلی مانند $g_2 \in SE(3)$ را با اســتفاده از تبدیل دیگری مانند $g_1 \in SE(3)$ از طریق ضرب ماتریس‌های تبدیل شان $T_1 . T_2 \in \mathbb{R}^{4\times4}$ در یک دیگر تبدیل کنیم تا تبدیل ترکیب شده $g_3 \in SE(3)$ را بدست بیاوریم:

$$\mathfrak{T}: SE(3) \times SE(3) \longrightarrow SE(3) . \boldsymbol{g_3} = \mathfrak{T}(g_1 . g_2) = T_1 T_2 \quad ۸\text{-}۳$$

که در نهایت g_3 به صورت زیر می‌باشد:

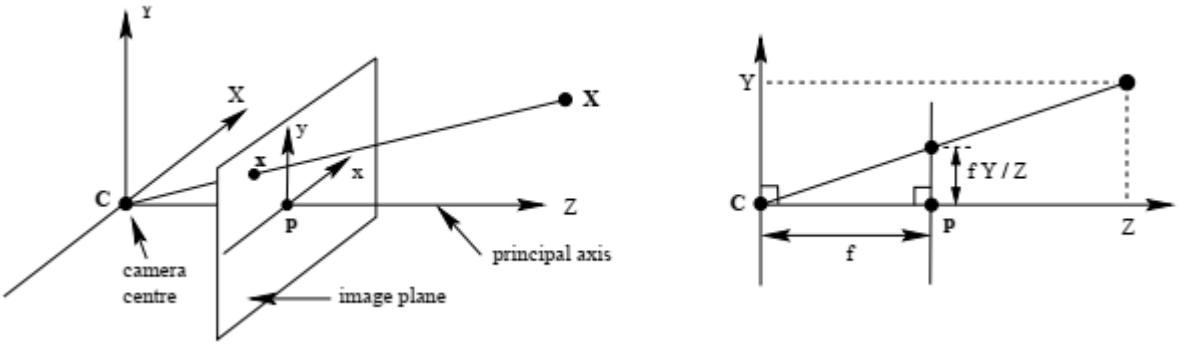
$$\boldsymbol{g_3} = \begin{bmatrix} R_1 & t_1 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R_2 & t_2 \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & R_1 t_2 + t_1 \\ 0^T & 1 \end{bmatrix} \in SE(3) \quad ۹\text{-}۳$$

تبدیل معکوس نیز به صورت زیر تعریف می‌شود

$$\mathfrak{T}^{-1}: SE(3) \times SE(3) \longrightarrow SE(3) . \boldsymbol{g_3} = \mathfrak{T}^{-1}(g_1 . g_2) = {T_1}^{-1} T_2 \quad ۱۰\text{-}۳$$

۳-۳ مدل دوربین

نگا شت نقاط سه بعدی یک منظره سه بعدی به روی یک صفحه ت صویر دو بعدی تو سط مدل دوربین تو صیف می شود، و این مدل تو سط یک ماتریس با ویژگی‌های خاص نمایش داده می شود. در این پایان‌نامه از مدل دوربین روزنه‌ای شـکل ۳-۱ اسـتفاده می‌کنیم. این مدل مرکز افکنش نقاط را بر روی یک صفحه منتصویر می‌کند، در جایی که مرکز دوربین C مرکز افکنش ا ست. د ستگاه مختصات دوربین، اقلید سی بوده و C در مرکز آن ا ست. صفحه ت صویر به و سیله $Z = f$ تعریف می شود که f فاصله کانونی است. محور اصلی، از C عبور کرده و به صفحه تصویر عمود است، محلی که محور اصلی، صفحه تصویر را قطع می‌کند نقطه اصلی[1] نامیده می‌شود $p = (p_x . p_y)^T \in R^2$.



شکل (۳-۱): مدل دوربین روزنه‌ای با مرکز C، نقطه اصلی p و فاصله کانونی f.[۱۳]

نقطه تصـویر دوبعدی $x = (u.v)^T \in \mathbb{R}^2$ که حالت سـه بعدی این نقطه $X = (X.Y.Z)^T \in \mathbb{R}^3$ ا ست به و سیله قطع دادن خط بین C و X با صفحه ت صویر تعیین می شود. با ا ستفاده از ت شابه مثلثاتی، نقطه ت صویر دوبعدی می‌تواند به صورت $x = (\frac{fX}{Z} . \frac{fY}{Z})^T$ محا سبه شود. از آنجاییکه نقطه اصلی معمولا در مرکز سیستم مختصات تصویر قرار ندارد، لازم است تا جابجایی نقطه اصلی نیز درنظر گرفته شود:

$$x = \left(\frac{fX}{Z} + p_x . \frac{fY}{Z} + p_y\right)^T \qquad ۳-۱۱$$

[1] Principal point

در مورد دوربین‌های CCD، مختصات تصویر به پیکسل است و در مختصات اقلیدسی اندازه‌گیری نشده است. علاوه بر این، پیکسل‌ها لزوما مربعی نیستند، که در این صورت فاکتورهای مقیاس را برای هر جهت دستگاه مختصات تصویر معرفی می‌کنیم. با داشتن m_x و m_y که تعداد پیسکل ها به ازای فاصله واحد در هر جهت را نشان می‌دهند، فاصله کانونی می‌تواند به صورت $f_x = fm_x$ و $f_y = fm_y$ در ابعاد پیکسل در جهت y و x بیان شود. مختصات نقطه اصلی در واحد پیکسل به صورت $c_x = m_x p_x$ و $c_y = m_y p_y$ است.

ماتریس کالیبراسیون دوربین که شامل پارامترهای داخلی دوربین است به صورت زیر بیان می‌شود:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad ۱۲\text{-}۳$$

نتیجه ضرب ماتریس کالیبراسون K در نقطه سه بعدی غیرهمگن $X = (x.y.z)^T \in \mathbb{R}^3$ متناظر با نقطه دوبعدی افکنش شده $\overline{x} \in \mathbb{R}^3$ در مختصات همگن است.

برای اینکه به طور مستقیم نقطه دوبعدی افکنش شده را در مختصات غیرهمگن بدست بیاوریم، تابع افکنش π را تعریف می‌کنیم. با استفاده از پارامترهای داخلی دوربین یک نقطه سه بعدی $X \in \mathbb{R}^3$ به یک نقطه دوبعدی $x \in R^2$ بر روی صفحه تصویر نگاشت می‌شود.

$$\pi: \mathbb{R} \times \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}^2 . \boldsymbol{x} = (u.v)^T = \pi(x.y.z) = \left(\frac{\boldsymbol{f_x x}}{\boldsymbol{z}} + c_x . \frac{\boldsymbol{f_x y}}{\boldsymbol{z}} + c_y\right)^{\boldsymbol{T}} \qquad ۱۳\text{-}۳$$

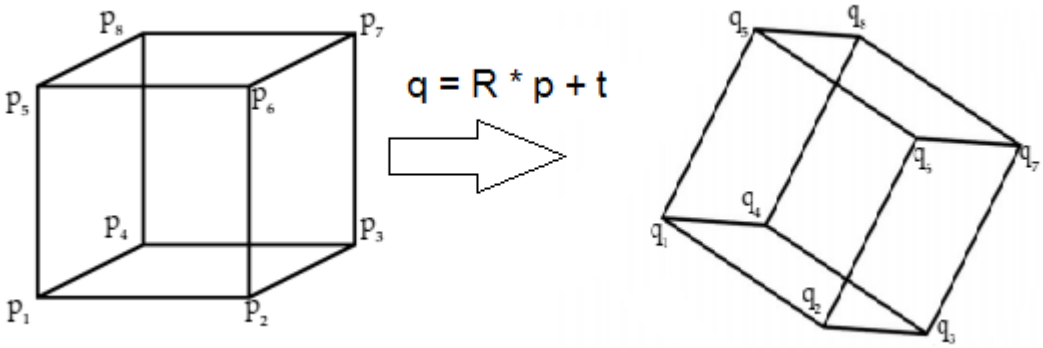
اگر مقدار عمقی مانند d برای یک نقطه دوبعدی $x = (u.v)^T \in \mathbb{R}^2$ داده شود، برای بازافکنش کردن این نقطه به مختصات سه بعدی $X \in \mathbb{R}^3$ در دستگاه مختصات دوربین[1] میتوانیم به صورت زیر عمل کنیم:

[1] Camera coordinate frame

$$\rho: \mathbb{R} \times \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}^3 . \boldsymbol{X} = (x.y.z)^T = \rho(u.v.d)$$
$$= \left(\frac{(\boldsymbol{u} - \boldsymbol{c_x})\boldsymbol{d}}{\boldsymbol{f_x}} . \frac{(\boldsymbol{u} - \boldsymbol{c_y})\boldsymbol{d}}{\boldsymbol{f_y}} . d \right)^{\boldsymbol{T}} \qquad ۱۴-۳$$

۳-۴ تخمین هم‌ترازی سه‌بعدی با استفاده از نقاط متناظر

مسـئله تعیین حرکت‌های جسـم صـلب برای ردیابی دوربین یکی از قسـمت‌های بسـیار مهم در بسـیاری از الگوریتم‌های SLAM اسـت. در روش‌های هم‌ترازی سـه‌بعدی بر مبنای ویژگی، ما معمولا نقاط سـه‌بعدی متناظر را در قاب‌های مختلف دوربین تعیین می‌کنیم، که به منظور تخمین حالت نسبی بین حالت‌های دو دوربین استفاده میشود مانند شکل ۳-۲



شکل (۳-۲): با داشتن دو مجموعه نقطه سه‌بعدی، ما می‌توانیم حالت نسبی بین آن‌ها را محاسبه کنیم

Registration ۱-۴-۳

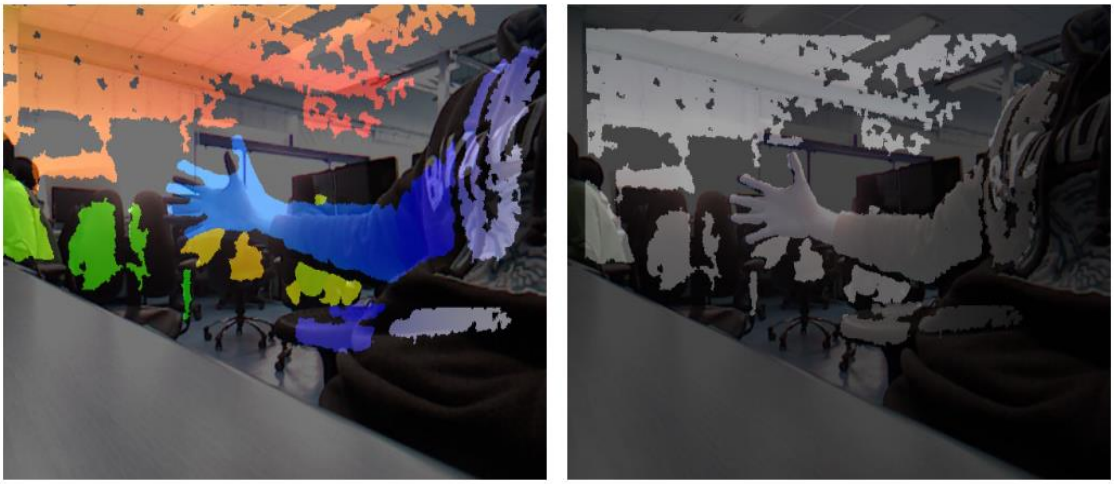
تثبیت تصـاویر یک قسـمت بنیادین در بینایی ماشـین اسـت. هدف از انجام این کار هم‌تراز کردن دو مجموعه داده است که از سیستم‌های مختصات متفاوتی بدست آمده‌اند. این مجموعه داده‌ها می‌توانند تصویر باشند، که در این مورد تثبیت تصاویر دوبعدی است یا می‌توانند مدل‌های سه بعدی یا ابرهای نقطه باشند، که در این صورت تثبیت تصاویر به صورت سه بعدی انجام می‌شود. بسته به نوع کاری که انجام می‌دهیم، تبدیل تخمین زده شده می‌تواند یک انتقال ساده یا یک تبدیل پیچیده(شامل انتقال و

چرخش) باشـــد. یک کاربرد شـــناخته شـــده تثبیت تصـــاویر، در نرم‌افزارهای پانوراما[1] اســـت که در دوربین‌های امروزی استفاده می‌شود، در این حالت چندین تصویر مختلف از محیط پیش‌رو به همدیگر متصل شده و یک تصویر پانوراما بزرگ را تشکیل می‌دهند. فرآیند تثبیت تصاویر برای تخمین موقعیت نسبی تصویرها و هم‌تراز کردن نواحی مشترک به کار می‌رود.

۳-۴-۲ تثبیت عمق

تثبیت تصاویر همانند آنچه در بالا گفته شد بر روی هم قرار دادن دو یا تعداد بیشتری تصویر از یک صحنه است، که در زمان‌های متفاوت، نمای متفاوت و/یا توسط سنسورهای مختلف گرفته شده باشد. این موضوع به صورت هندسی دو تصویر را بر روی هم منطبق می‌کند. شکل ۳-۳ تفاوت بین داده‌های تثبیت شده و تثبیت نشده را نشان می‌دهد. در این بخش فرآیند تثبیت کردن از دو سنسور متفاوت را بیان می‌کنیم (دوربین رنگی و مادون قرمز). نمای دید این دو سنسور تقریبا موازی است ولی به طور افقی جابه‌جا شده‌اند. با داشتن ماتریس‌های K_{ir} (ماتریس پارامترهای داخلی سنسور مادون قرمز) و K_{RGB}(ماتریس پارامترهای داخلی دوربین رنگی)، گردش R و انتقال t بین دو دوربین ما قادر خواهیم بود تا مقادیر عمق را که مبداشان در سیستم مختصات دوربین مادون قرمز قرار دارد به سیستم مختصات دوربین رنگی افکنش کنیم. بعد از این کار قادر خواهیم بود که عمق مورد نظرِ پیسکل رنگی متناظر را بدست آوریم.

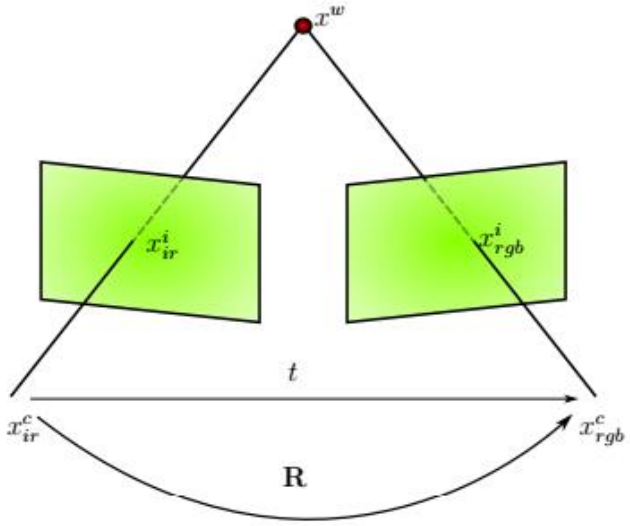
[1] Panorama



(a) Unregistered depth (b) Registered depth

شکل (۳-۳): تثبیت عمق- تصویر سمت چپ تصویری را همراه با اطلاعات عمق نشان می‌دهد که بر روی هم قرار گرفته‌اند و رجیستر نشده اند. مقادیر عمق با اطلاعات تصویر رنگی متناظر نبوده و برابر نیستند. تصویر سمت راست مدل تثبیت شده تصویر رنگی و عمق است که مقادیر عمق تصویر رنگی را می‌توان بدست آورد[۱۵].

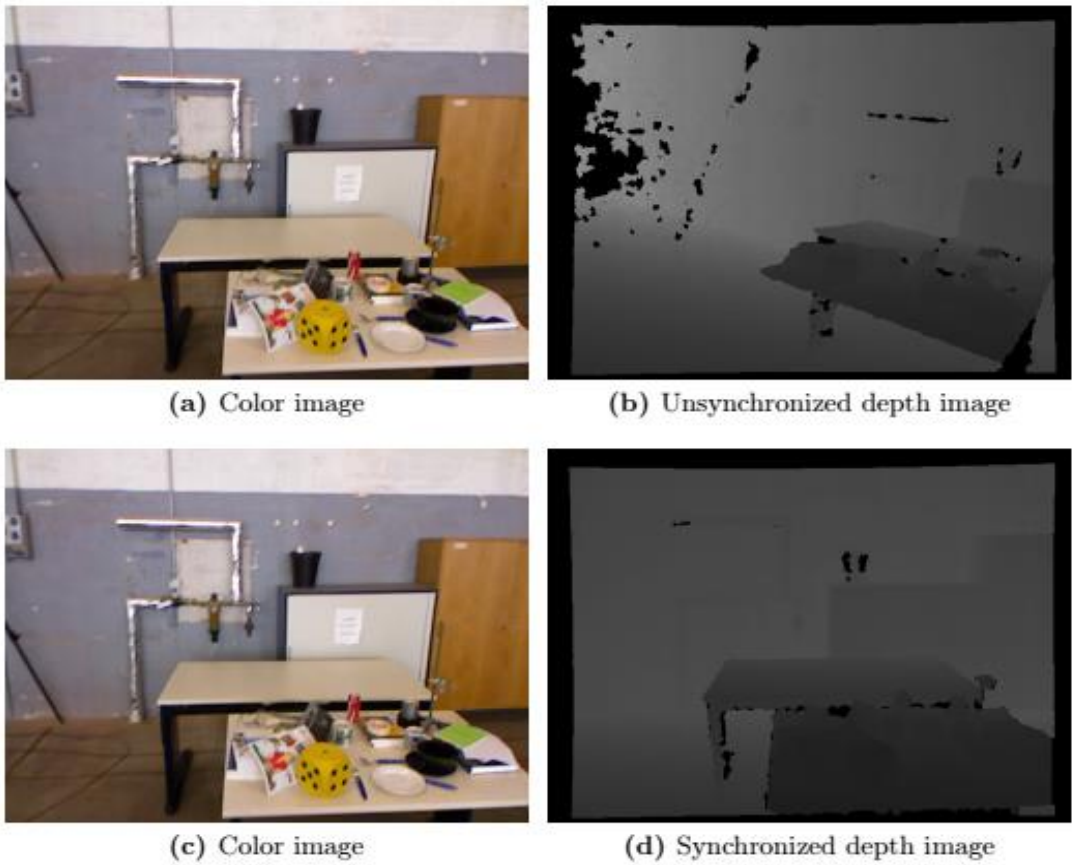
این روش برای بدست آوردن حرکت که در بخش ۳-۲ صحبت شد مهم است. زیرا اگر مقادیر عمق‌ها اشتباه باشند ما نمی‌توانیم به طور صحیح حرکت را بین دو قاب متوالی بدست آوریم. شکل ۳-۴ مسئله تثبیت کردن تصویر را به صورت گرافیکی نمایش می‌دهد.



شکل (۳-۴): رجیستر کردن عمق و رنگ

۳-۴-۳ هم‌زمانی

اگر مقادیر عمق و ت صاویر رنگی با هم همزمان نبا شند برای فرآیند تثبیت که در بخش قبل گفته شد دچار مشکل می‌شویم. دوربین‌های RGB-D معمولا زمان بیشتری نیاز دارند تا تصویر عمق را تولید کنند. در دوربین کینکت مایکروسافت اطلاعات عمق در حدود ۲۰ میلی‌ثانیه بعد از دریافت اطلاعات رنگی زمان می‌برد تا بد ست آید. این عمل خود باعث می شود تا اطلاعات عمق و رنگ با هم هماهنگ نباشند و عمقی که برای تصویر رنگی فعلی بدست می‌آوریم اشتباه باشد. البته این مشکل برای دوربین‌هایی که ثابت بوده و حرکت نسبتا آرامی دارند زیاد مشکل‌ساز نیست ولی هرچه سرعت حرکت دوربین بیشتر شود این مسئله خود را بیشتر نشان می‌دهد. این مشکل در شکل ۳-۵ نشان داده شده است. البته ما این مشکل را توسط کتابخانه OpenNI برطرف کرده‌ایم.



شکل (۳-۵): مقایسه بین تصاویر رنگی و عمق هم زمان و غیر هم زمان. این تصاویر نمایش دهنده مسئله هم زمانی هستند. (الف)در حالی که تصاویر بالا تصویر رنگی نشان می‌دهد ولی(ب) عمق دریافت شده که به وسیله دوربین RGB-D ضبط شده است و متعلق به چند تصویر رنگی قبل‌تر است. (پ) تصویر رنگی و(ت) تصویر عمق هم زمان آن که بر یکدیگر منطبق هستند[۱۵].

۳-۴-۴ تثبیت سه‌بعدی

به طور ویژه دقت کل سامانه باز سازی سه بعدی به و سیله تخمین موقعیت‌های نسبی دوربین[1] تعیین می‌شود. یک روش شناخته شده غیر تکراری برای تخمین تبدیل اقلیدسی سه بعدی از تناظرهای نقطه سه بعدی[2] که در [۱۶] بیان شده است به صورت زیر است

دو مجموعه نقطه $Q = \{q_i\}$ و $P = \{p_i\}$ با $n \geq 4$ و $i \in 1 \dots n$ برای هر ابر نقطه، داده شده است. نقطه $q_i \in \mathbb{R}^3$ و $p_i \in \mathbb{R}^3$ نقاط سه بعدی متناظر هستند و با نویز N_i ذخیره شده در یک بردار N ترکیب شده‌اند. در این بخش فرض میکنیم که تمام نقاط متناظر بین این دو مجموعه نقطه صحیح می‌باشند. اگر خطا وجود دارد استفاده از روش‌های مبتنی بر RANSAC ضروری است.

این دو مجموعه نقطه به وسیله تبدیل $\hat{g} \in SE(3)$ به هم مرتبط می‌شوند، که به وسیله یک ماتریس $R \in \mathbb{R}^3$ و یک بردار $t \in \mathbb{R}^3$ نمایش داده می‌شود

$$q_i = Rp_i + t + N_i \qquad ۱۵-۳$$

برای تعیین چرخش بهینه $\hat{R}$ و انتقال بهینه $\hat{t}$، ما باید مسئله حداقل مربعات زیر را کمینه کنیم

$$\Sigma^2 = \sum_{i=1}^{n} \left\| q_i - \left(\hat{R}_{P_i} + \hat{t}\right) \right\|^2 \qquad ۱۶-۳$$

الگوریتمی که برای پیدا کردن چرخش و انتقال بهینه استفاده می‌شود به صورت زیر است:

مجموعه نقطه مرکز صفر[3]: به منظور محاسبه چرخش، ابتدا مراکز(مرکزهای توده) $\bar{q}$ و $\bar{p}$ دو مجموعه نقطه تعیین می‌شوند:

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i \quad and \quad \bar{q} = \frac{1}{n}\sum_{i=1}^{n} q_i \qquad ۱۷-۳$$

[1] Relative camera poses
[2] 3D point correspondences
[3] Zero-centered point set

هر دو مجموعه نقطه به واسطه کم کردن مرکز از هر نقطه به مرکز دستگاه مختصات متناظرشان منتقل می‌شوند. در نتیجه ما نقطه q_{ci} و p_{ci} مرکز صفر را بدست می‌آوریم:

$$p_{c_i} = p_i - \bar{p} \quad and \quad q_{c_i} = q_i - \bar{q} \qquad ۱۸-۳$$

ماتریس چرخش بهینه $\hat{R}$: به منظور پیدا کردن ماتریس $\hat{R}$ که به خوبی بتواند دو مجموعه نقطه مرکز صفر را بر هم منطبق کند، مسئله کمینه کردن حداقل مربعات[1] را دوباره بازنویسی می‌کنیم[۱۶]:

$$\begin{aligned} \Sigma^2 &= \sum_{i=1}^{n} \left\| q_{c_i} - \hat{R} p_{c_i} \right\|^2 = \sum_{i=1}^{n} \left(q_{c_i} - \hat{R} p_{c_i} \right)^t \left(q_{c_i} - \hat{R} p_{c_i} \right) \\ &= \sum_{i=1}^{n} \left\| {q_{c_i}}^T q_{c_i} + {p_{c_i}}^T p_{c_i} - 2 {q_{c_i}}^T \hat{R} p_{c_i} \right\| \end{aligned} \qquad ۱۹-۳$$

کمینه کردن معادله ۳-۱۹ میتواند به وسیله بیشینه کردن عبارتی که شامل $\hat{R}$ است بدست بیاید. اگر داشته باشیم[۱۶]:

$$F = \sum_{i=1}^{n} {q_{c_i}}^T \hat{R} p_{c_i} = Trace\left(\sum_{i=1}^{n} \hat{R} p_{c_i} {q_{c_i}}^T \right) = Trace(\hat{R} H) \qquad ۲۰-۳$$

به صورتی که:

$$H = \sum_{i=1}^{n} p_{c_i} {q_{c_i}}^T \qquad ۲۱-۳$$

با اســتفاده از تجزیه مقدارهای منفرد(SVD[2]) $H = U \sum V^T$، تریس[3] با قرار دادن ماتریس چرخش به صورت زیر بیشینه خواهد شد

[1] Least-squares minimization
[2] Singular value decomposition
[3] Trace

$$\hat{R} = VU^T \qquad ۲۲-۳$$

اگر دترمینان $det(\hat{R})$ یک باشـــد، آنگاه حل منحصـــر به فرد اســـت و ما چرخش بهینه را پیدا کرده‌ایم. اگر $det(\hat{R})$ یک منفی باشـد، آنگاه $\hat{R}$ از منظر عددی درسـت بوده ولی می‌گویم رفلکشـن[1] رخ داده است ، این اتفاق می‌تواند زمانی که نویز زیاد یا مجموعه نقاط هم صفحه[2] هستند رخ دهد. در این حالت چرخش به صـورت $\hat{R} = V'U^T$ محاسـبه شـود، که V' با تغییر دادن علامت سـتون سـوم بردار V بدست می‌آید.

انتقال بهینه $\hat{t}$: یکبار که $\hat{R}$ تعیین شد، تبدیل بهینه $\hat{t}$ در نهایت به صورت تفاضل بین مرکز $\bar{p}$ با مرکز $\bar{q}$ محاسبه می‌شود:

$$\hat{t} = \bar{p} - \hat{R}\bar{q} \qquad ۲۳-۳$$

همانظور که ما چرخش و انتقال بهینه را تعیین کردیم، همچنین حالت نســـبی بهینه $\hat{g} = (\hat{R}.\hat{t}) \in SE(3)$ بین دو مجموعه نقطه سه بعدی را بدست آورده‌ایم

۳-۵ کالیبراسیون دوربین

کالیبراسیون فرآیند بدست آوردن پارامترهای مختلف دوربین است. کارخانه سازنده ممکن است این پارامترها را در اختیار ما بگذارد ولی به این علت که ممکن اســت برای بازســازی ســه‌بعدی دقیق نباشد از کالیبراسیون استفاده می‌کنیم.

فرآیند کالیبراسیون فعال[3] به وسیله نمایش الگوهایی شناخته شده به دوربین و تحلیل تصویرهای بدست آمده، صورت می‌گیرد. برای کالیبراسیون دوربین، ایده اولیه این است که مجموعه‌ای از نقاط محیط را نشان دهیم به طوری که موقعیت سه‌بعدی آن‌ها شناخته شده باشد. سپس نیاز است تا

[1] Reflection
[2] Co-planar
[3] Active calibration

بدانیم کجا این نقاط بر روی تصـویر مشـاهده شـده اسـت. از معادله افکنش می‌توانیم این پارامترها را بد ست آوریم. برای بد ست آوردن پارامترهای دقیق نیاز ا ست تا تعداد زیادی از این نقاط را م شاهده کنیم.

راه‌کارها:

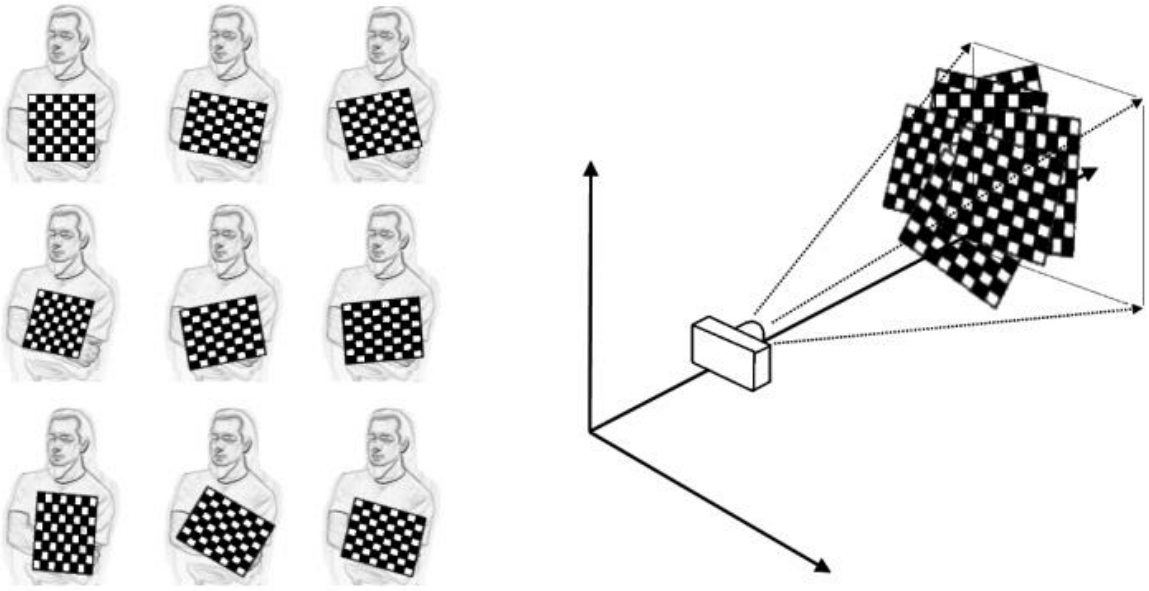
۱-یک تصویر از محیط با تعداد زیادی نقطه سه‌بعدی بگیریم که این کار تقریبا غیر ممکن است.

۲-از یک سری نقطه سه‌بعدی چندین تصویر از زوایا مختلف گرفته شود.

راه دوم سـاده‌تر اسـت ولی نیازمند این اسـت تا موقعیت هر نمای[1] دوربین به اضـافه پارامترهای داخلی محاسـبه شـوند. OpenCV[۱۷] از یک الگوی شـطرنجی به منظور تولید یک مجموعه نقاط سه‌بعدی محیط برای کالیبرا سیون ا ستفاده می‌کند. این الگو نقاط را در گو شه هر مربع می سازد و از آنجا که این الگو صاف است فرض می‌کنیم $Z = 0$ است. این تابع به صورت خودکار گوشه‌های این الگوی شطرنجی را شنا سایی می‌کند. خروجی این تابع مخت صات نقاط درونی شنا سایی شده الگوی نشان داده شده است.

به منظور اجرای کالیبراسـیون نیاز داریم تا نقاط سـه بعدی متناظر را مشـخص کنیم. این نقاط را می‌توانیم در واحد دلخواهمان مشـخص کنیم(سـانتی‌متر یا اینچ). با این وجود سـاده‌ترین روش برای انجام این کار این ا ست که فرض کنیم هر مربع یک واحد ا ست. بنابراین اگر الگوی شطرنجی به ابعاد ۷ × ۵ مربع داشـته باشـیم اولین نقطه (۰و۰و۰) و آخرین آن (۰و۴و۶) اسـت که در کل، ۳۵ نقطه می‌شود. ولی این تعداد نقطه برای کالیبراسیون کم است. بنابراین نقاط بیشتری نیاز داریم که با گرفتن تصـویر از زوایای مختلف بدسـت می‌آید. برای این کار می‌توانیم یا دوربین را جابه‌جا کنیم یا مکان تصویر شطرنجی را تغییر دهیم که از منظر ریاضی هر دو یکسان هستند. (شکل ۳-۶)

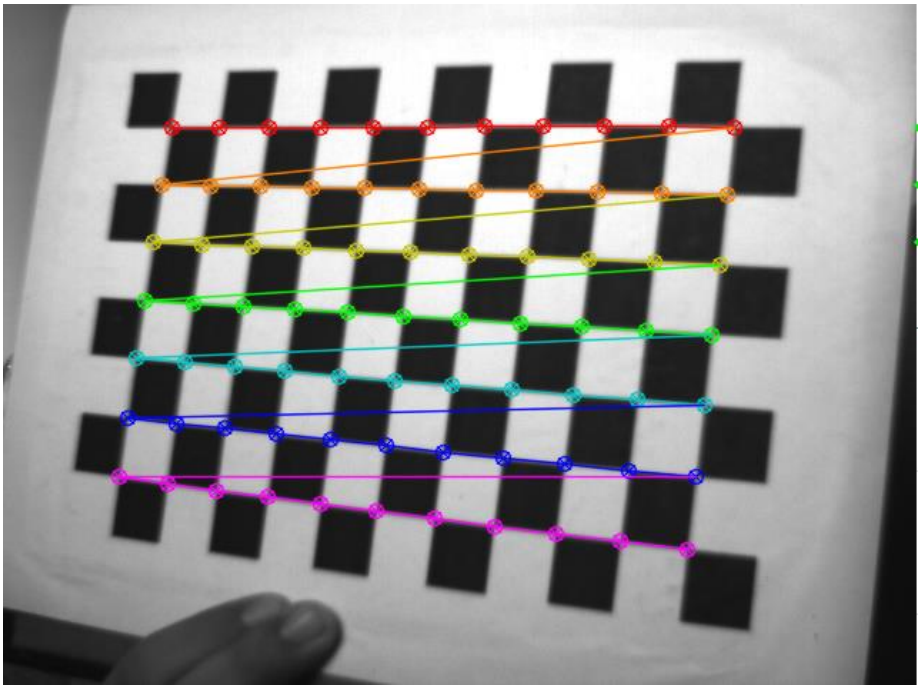
[1] View



شکل (۳-۶): (سمت چپ) تصویرهای از یک صفحه شطرنجی که در جهت‌های مختلف نگاه داشته شده است، اطلاعات کافی برای حل کامل موقعیت‌های آن تصویرها در مختصات جهانی (که وابسته به دوربین هستند) و پارامترهای داخلی دوربین را فراهم می‌کند[۱۸].

۳-۵-۱ ترسیم گوشه‌های صفحه شطرنجی

به طور خاص زمانی که ما برنامه را عیب‌یابی می‌کنیم، معمولا مطلوب ا ست که گو شه‌های صفحه شطرنجی را بر روی تصویر رسم کنیم. به این طریق می‌توانیم ببینیم که آیا گوشه‌های افکنش شده با گو شه‌های م شاهده شده صفحه شطرنجی منطبق ه ستند یا نه. به منظور انجام این کار $OpenCV$ تابعی را فراهم کرده اســـت که این موضـــوع را برای ما مدیر یت می‌کند. اگر تمام گوشـــه پیدا نشد،گوشه‌های پیدا شده توسط دایره‌های کوچک قرمز رنگ مشخص می‌شوند. اگر تمام الگو شناسایی شد، بنابراین گو شه‌ها با رنگ‌های مختلف تر سیم می شوند(هر سطر رنگ خاص خودش را دارد) و به وسیله خط‌هایی به هم متصل شده که ترتیب گوشه شناسایی شده را نشان می‌دهد.(شکل ۳-۷)



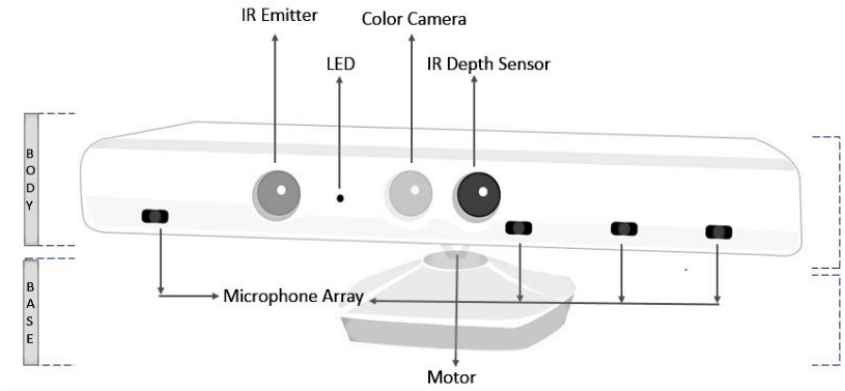
شکل (۳-۷): نتیجه استفاده از drawChessboardCorners. یکبار که شما به وسیله findChessboard گوشه‌ها را پیدا کردید، شما می‌توانید جاییکه این گوشه‌ها پیدا شده‌اند آن‌ها را ترسیم کنید(دایره‌های کوچک بر روی گوشه‌ها).

بعد از انجام فرآیند کالیبراسیون برای دوربین رنگی کینکت، مقادیر ماتریس K به شرح زیر است:

$$K = \begin{bmatrix} 517.3 & 0 & 318.6 \\ 0 & 516.5 & 245.3 \\ 0 & 0 & 1 \end{bmatrix}$$

۳-۶ سنسور کینکت

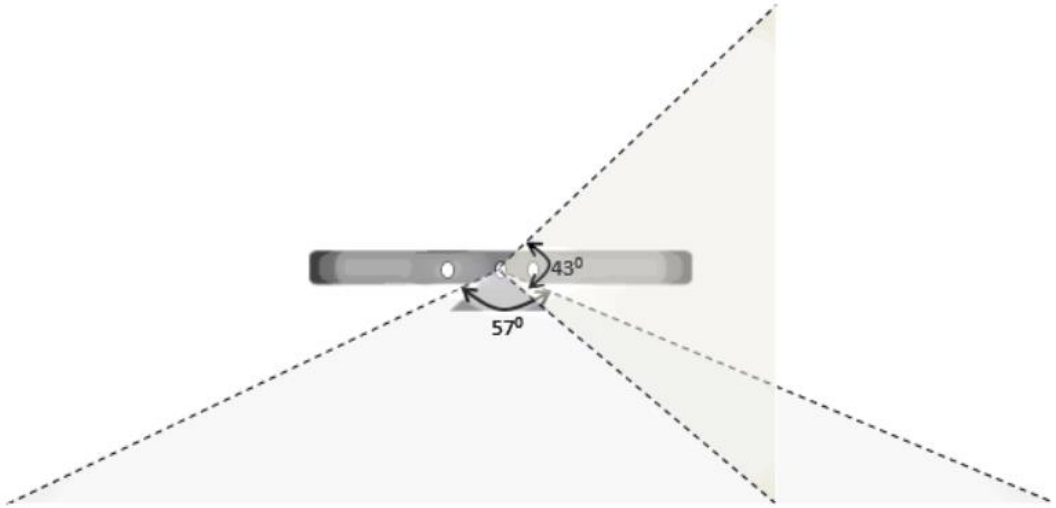
سنسور کینکت مایکروسافت یک دستگاه الکترونیکی است که از سنسورهای عمق، دوربین رنگی، یک چراغ LED، موتور و یک مجموعه آرایه میکروفنی تشکیل شده است(شکل ۳-۸).



شکل (۳-۸): اجزای تشکیل دهنده دستگاه کینکت

۳-۶-۱ دوربین رنگی

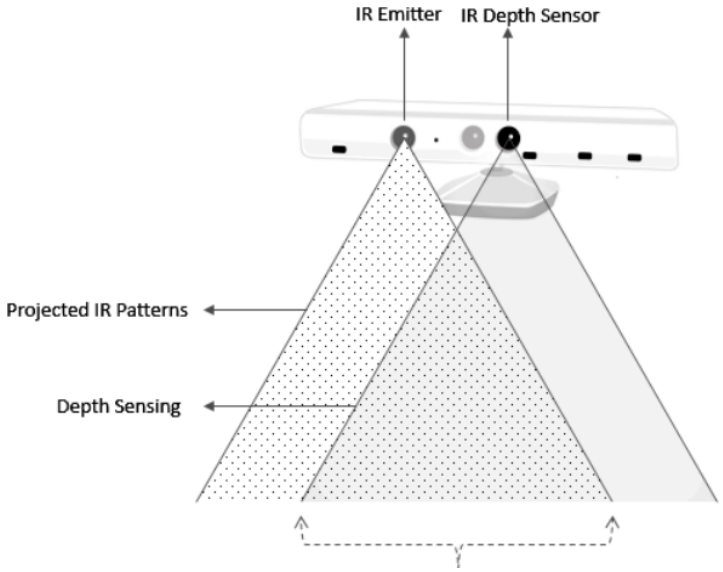
دوربین رنگی کینکت مسئول ضبط و دریافت داده‌های ویدئو رنگی است. این سنسور مقادیر آبی، قرمز و سبز محیط اطراف را دریافت می‌کند. دوربین رنگی کینکت داده‌ها را با نرخ ۳۰ قاب بر ثانیه در دقت ۴۸۰ × ۶۴۰ دریافت می‌کند، و حداکثر دقت این دوربین ۹۶۰ × ۱۲۸۰ است که در این حالت با نرخ ۱۲ قاب بر ثانیه اطلاعات را دریافت می‌کند. محدوده دید افقی و عمودی این دوربین به ترتیب ۵۷ و ۴۳ درجه می‌باشد(شکل ۳-۹).



شکل (۳-۹): محدوده دید افقی و عمودی دوربین رنگی کینکت

۳-۶-۲ فرستنده و سنسور عمق مادون قرمز

سنسور عمق کینکت از یک فرستنده و سنسور عمق مادون قرمز تشکیل شده است. این دو سنسور با هم کار می‌کنند تا بتوانند عمق را استخراج کنند. فرستنده مادون قرمز از نظر ظاهری شبیه به یک دوربین است ولی در حقیقت یک پروژکتور مادون قرمز می‌باشد که به صورت مداوم الگویی را به محیط اطراف خود ساطع می‌کند. این الگو به صورت نقاطی هستند که با چشم قابل دیدن نبوده ولی سنسور عمق مادون قرمز این الگو را دریافت می‌کند. سنسور عمق مادون قرمز این الگو دریافت شده را با الگو فرستاده شده مقایسه کرده و مقادیر عمق را بدست می‌آورد.(شکل ۳-۱۰)



شکل (۳-۱۰): نمایش الگو فرستاده شده توسط فرستنده مادون قرمز و دریافت آن توسط سنسور عمق

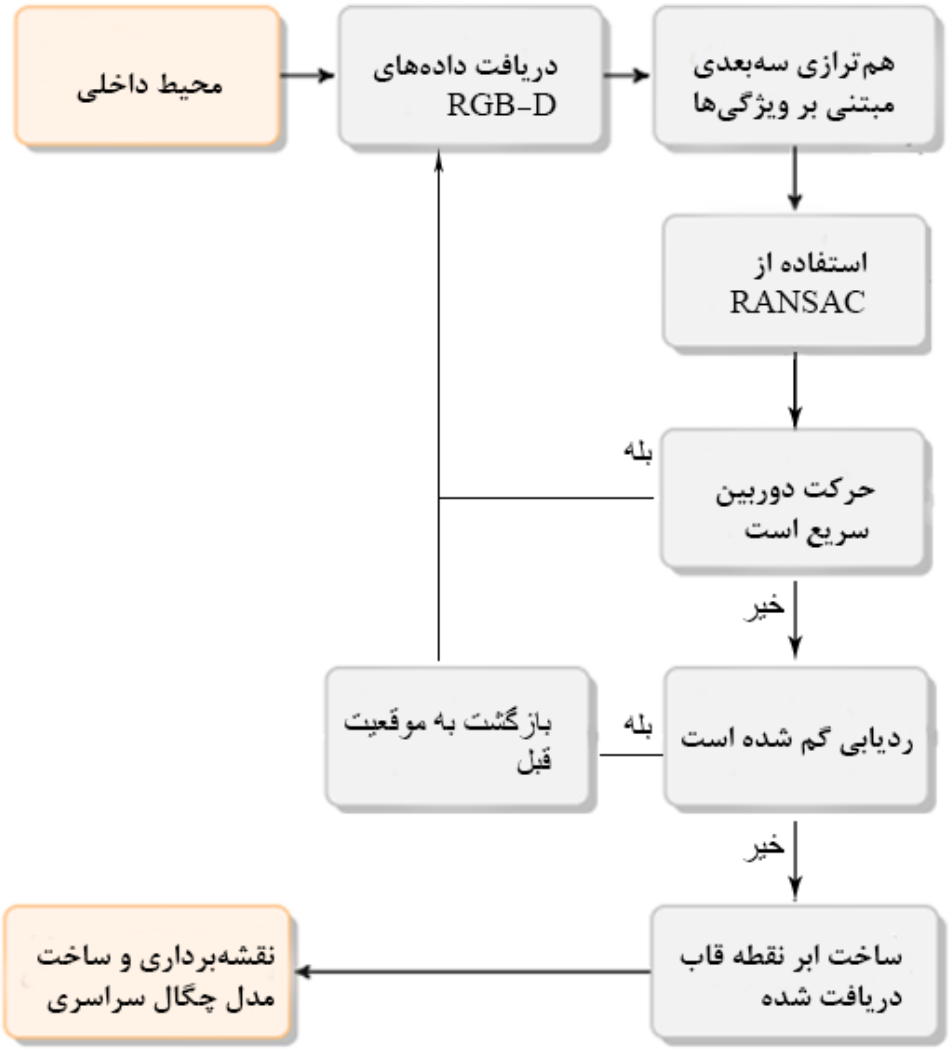
فصل چهارم: بازسازی سه‌بعدی

۴-۱ مقدمه

بعد از بیان مباحث مورد نیاز برای بازسازی سه بعدی در فصل قبل، در این فصل، روندنمای سیستم بازسازی سه بعدی RGB-D برای بازسازی محیط داخلی به طور کامل مطرح می‌شود.

۴-۲ راه‌کار پیشنهادی برای بازیابی سه بعدی

برای باز سازی سه بعدی محیط داخلی، روندنمایی مانند [۹] توسعه داده شده است. شکل ۴-۱ اجزای متفاوت این روش را نشان میدهد.



شکل(۴-۱): نمای سیستم توسعه داده شده برای این پایان نامه

ابتدا، ما داده‌های RGB محیطی را که می‌خواهم نقشه برداری کنیم، از سنسور RGB-D دریافت می‌کنیم. شناسایی و استخراج ویژگی‌های دوبعدی بصری از تصویر رنگی به ما اجازه می‌دهد تا تصویرها را دو به دو باهم مقایسه کنیم. این تناظرهای دوبعدی بدست آمده به اطلاعات عمق اضافه می‌شوند و در نتیجه نقاط سه‌بعدی متناظر را بدست می‌آوریم. با توجه به آنچه که در بخش تثبیت سازی سه‌بعدی فصل قبل گفته شد می‌توانیم برای محاسبه هم‌ترازی سه‌بعدی نسبی[1] بین قاب‌ها اقدام کرده و از آنجا که نقاط متناظر سه‌بعدی ما الزاما بدون خطا نمی‌باشند با استفاده از RANSAC[2]، بهینه شده و ماتریس تبدیل مناسب بین قاب‌ها را بدست می‌آوریم. با داشتن این موقعیت‌های نسبی دوربین برای هر جفت قاب، حالت مطلق هر قاب دوربین با توجه به دستگاه مختصات جهانی[3] می‌تواند تعیین شود. بعد از آنکه تبدیل نسبی بین دو قاب بدست آمد میزان حرکت دوربین و ردیابی ویژگی‌ها را بررسی می‌کنیم. سپس ابر نقطه هر فریم ساخته می شود. در نهایت این ابر نقاط بدست آمده به فرآیند نقشه برداری اضافه می‌شوند. جزئیات کامل هر قسمت در ادامه توضیح داده می‌شود.

۴-۳ بدست آوردن داده‌های سه بعدی

در هر سیستم بازسازی سه بعدی، اولین گام بدست آوردن داده‌های محیط مورد نظر است. همانطورکه قبلا ذکر شد، برای بدست آوردن داده‌های RGB-D به منظور ساخت مدل سه بعدی یک محیط ما از سنسورهای RGB-D مانند کینکت استفاده می‌کنیم.

در داده‌های RGB-D، هر قاب شامل یک تصویر رنگی RGB با نقشه عمق متناظر آن تصویر رنگی است. در یک تصویر رنگی، ما می‌توانیم رنگ را برای یک نقطه دوبعدی $(u.v)^T$ با استفاده از تابع زیر بدست آوریم:

$$I_{RGB}: \mathbb{R}^2 \longrightarrow \mathbb{R}^3 . (r.g.b)^T = I_{RGB}(u.v) \qquad ۱-۴$$

[1] Relative 3D alignment
[2] RANdom Sample Consensus
[3] Global world coordinate frame

نقشه عمق یک تصویر دوبعدی است، که شامل مقادیر عمق محاسبه شده توسط سنسور است. عمق متناظر برای یک نقطه دوبعدی به وسیله تابع زیر پیدا می‌شود:

$$I_{RGB}: \mathbb{R}^2 \longrightarrow \mathbb{R}. z = I_d(u.v) \qquad ۲-۴$$

با توجه به آنچه که در فصل سه گفته شد، یک تناظر یک به یک بین پیکسل‌ها در تصویر رنگی و مقادیر عمق آن‌ها وجود دارد به این معنی که رنگ و عمق تصویرها هم‌تراز[1] هستند.

با استفاده از پارامترهای داخلی بدست آمده برای دوربین، هر پیکسل میتواند در دستگاه مختصات سه‌بعدی دوربین با استفاده از معادله ۳-۱۴ بازافکنش شود. این کار به ما اجازه می‌دهد که یک ابر نقطه سه‌بعدی تماما رنگی بر مبنای عمق تشکیل بسازیم.

۴-۳-۱ سنسورهای RGB-D

روش پیشنهادی ما برای سنسورهای مختلف، مادامی که داده‌های RGB-D را فراهم کنند مناسب است. از این رو این امکان وجود دارد تا برای یک برنامه خاص از چند اسکنر سه‌بعدی استفاده کنیم.

نقشه‌های پراکنده، عمق را فقط برای همان تعداد اندکی از نقاط کلیدی که شناسایی شدند دارا است؛ در نتیجه برای بازسازی چگال، دقیق و کامل مدل سه بعدی مناسب نیستند. بنابراین لازم است نقشه عمق چگال را بدست آوریم به طوری که شامل مقادیر عمق برای هر پیکسل در هر قاب باشد. نقشه عمق می‌تواند از تصویرهای یک یا چندین دوربین CCD با استفاده از الگوریتم‌های مناسب محاسبه شود. در حالی که پیشرفت‌های مهمی در ساخت نقشه‌های عمق سه بعدی از یک دوربین استاندارد دستی با استفاده از چندین نما در کارهایی مانند[۷, ۱۹] صورت گرفته است، مسئله تعیین عمق در واحد واقعی همچنان به صورت یک چالش، باقی مانده است.

[1] pre-registered

برای بدست آوردن نقشه عمق مورد نیاز، دستگاه‌های حسگر عمق مانند دوربین‌های ToF[1]، اسنکرهای سه بعدی یا دوربین‌های RGB-D می‌توانند استفاده شوند. تمام این دستگاه‌ها مقادیر دقیق عمق را به صورت واقعی تولید می‌کنند. در حالی که دو نوع اول هنوز از نظر قیمتی گران هستند، وجود سنسور RGB-D ارزان قیمت کینکت زمینه جدیدی از باز سازی سه بعدی را برای پژوهشگران گشوده است. کینکت مایکروسافت، که در [۲۰] استفاده شده است، همانطور که در فصل قبل گفته شد، تصاویر رنگی را همراه با نقشه عمق در ابعاد ۶۴۰ × ۴۸۰ پیکسل به صورت آنی در نرخ قاب ۳۰ قاب بر ثانیه فراهم می‌کند. در سنسورهای RGB-D، روش نور ساختاریافته[2] [۲۱] برای تولید آنی نقشه چگال عمق استفاده شده است. اگرچه این داده‌های عمق، کیفیت بالایی دارند ولی هنوز تحت تاثیر نویز قرار می‌گیرند و در برخی موارد، مقادیر عمق مفقود می‌شود. به علاوه، مقادیر عمق فقط تا فواصل محدودی که معمولا کمتر از ۵ متر است، قابل محاسبه است.

۴-۳-۲ مزیت استفاده از کینکت برای SLAM

در این پایان‌نامه همانطور که در فصل قبل اشاره شد از سنسور کینکت مایکروسافت xbox ۳۶۰ استفاده شده است. البته سنسورهای مشابهی مانند ASUS Xtion Pro Live[22] نیز موجود هستند که فرآیند باز سازی سه‌بعدی را انجام می‌دهند(شکل ۲-۴). برای ارتباط با این سنسور از کتابخانه[3] OpenNI[4] استفاده می‌کنیم. تصاویر RGB و نقشه عمق که توسط OpenNI برای ما فراهم می‌شوند از قبل رجیستر شده بوده و به راحتی این امکان را می‌دهند که مقدار عمق برای هر پیکسل تصویر رنگی را بدست بیاوریم. در حالی که پارامترهای داخلی دوربین برای این سنسورها از قبل موجود هستند ولی ما میتوانیم این پارامترها را با استفاده از یک کتابخانه کالیبراسیون دوربین مانند نوار ابزار کالیبراسیون متلب[۲۳] یا با استفاده از روش گفته شده در بخش ۳-۵ بدست آوریم.

[1] Time of Flight
[2] structured light
[3] Framework
[4] Open natural interface

شکل (۴-۲): سنسور RGB-D Asus Xtion Pro [17] و سنسور Microsoft Kinect.[۲۴]

کینکت مسئله SLAM را به طور قابل ملاحظه‌ای ساده کرده و بنابراین یک سنسور عالی برای مسیریابی ربات است. کینکت الگوریتم SLAM را از دو طریق بهبود می‌بخشد:

همانطورکه گفته شد دیگر نیازی به حدس اولیه عمق برای یک ویژگی تازه مشاهده شده نیست چون که پیکسل‌های تصویر کینکت دارای اطلاعات عمق هستند.

به جای آنکه فقط ویژگی‌های پراکنده را به نقشه اضافه کنیم می‌توانیم ابر نقطه کاملی را تولید کرده و به نقشه مورد نظر اضافه کنیم و این ابرنقطه می‌تواند اکثر جزئیات محیط را در خود داشته باشد.

۴-۳-۳ فرایند دریافت و پردازش نقشه عمق

از آنجا که دقت اندازه‌گیری عمق سنسورهای سه بعدی به فاصله سنسور تا سطح اندازه‌گیری شده بستگی دارد، ما فقط اندازه‌هایی از عمق را که در محدوده‌ی خاصی هستند برای پردازش‌های بعدی استفاده می‌کنیم. اگر مقدار عمق بدست آمده از آستانه‌ی مورد نظر بیشتر شد، عمق آن نقطه را صفر در نظر می‌گیریم، به عبارتی فرض می‌کنیم عمق برای این پیکسل موجود نیست. در نظر نگرفتن

مقادیری از فاصله‌ها که عدم قطعیت در آن‌ها بالاست دقت تطبیق سه بعدی قاب[1] را افزایش داده و در نهایت دقت بازسازی سه بعدی نهایی را افزایش می‌دهد.

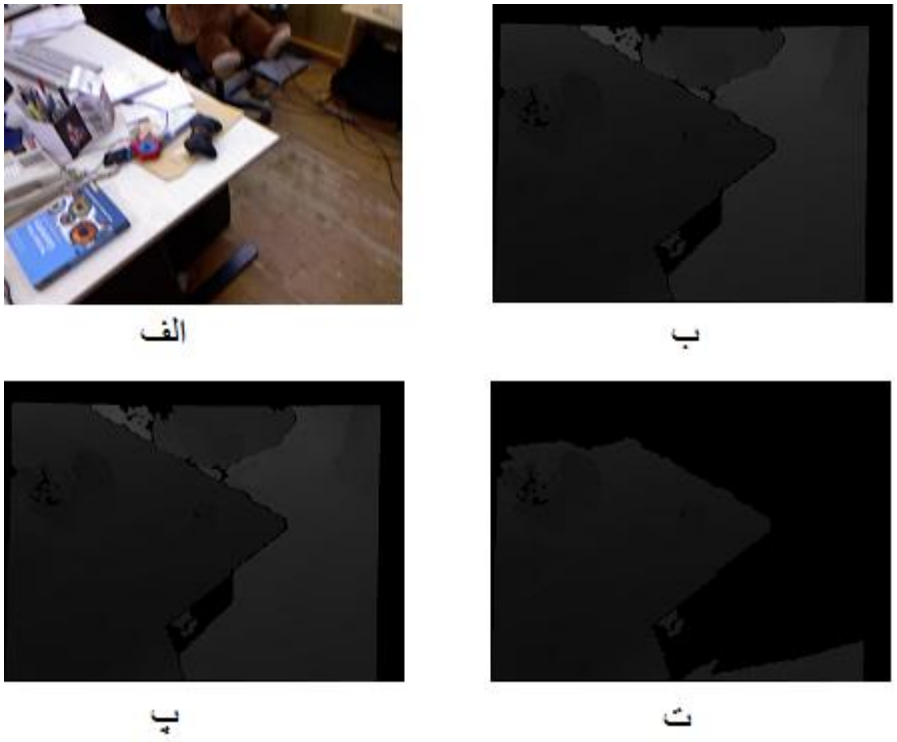
فرایند واقعی بدســـت آوردن داده‌ها به وســـیله ضـــبط قاب‌های متوالی از محیط داخلی و با تغییر حالت دوربین صورت می‌گیرد. در پژوهش ما، داده‌ها با یک سنسور RGB-D که در دست نگاه داشته شده است ضبط می‌گردند. انجام این کار بسیار ساده بوده و شبیه خیلی از تحقیقات حال حاضر است [۲۰].

برای مسائل بازسازی کاملا خودکار، می‌توان دوربین را بر روی یک بازوی رباتیک قرار داده و آن را بر طبق یک مسیر از پیش تعیین شده حرکت دهیم. مشکل اصلی این است که هنگامی که داده را ضبط می‌کنیم، سنسور را در فاصله معقولی از سطح مورد نظر نگاه داریم. اگر خیلی از سطح مورد نظر دور باشیم، فقط جزئیات کمی در تصویر رنگی نمایان شده و ضبط می‌گردند. با این وجود اگر سنسور خیلی به شی نزدیک باشد، آنگاه ردیابی دوربین با مشکل روبرو می‌شود، چرا که سنسورهای عمق که بر اســاس نور ســاختاریافته کار می‌کنند یک حداقل فاصــله برای تعیین مقادیر عمق دارند. برای اینکه بتوانیم کیفیت خوبی برای ویژگی‌های یافت شده، هم‌ترازی سه‌بعدی و بازرسی بصری بدست آوریم، ما سنسور را در فاصله بین ۰٫۸ متر تا ۳٫۰ متر از سطح یا محیط مورد نظر نگاه می‌داریم (شکل ۴-۳).

[1] 3D frame alignment

شکل (۴-۳): داده‌های RGB-D که توسط سنسور RGB-D در فاصله بین ۰،۸ متر و ۳،۰ متر از سطح بدست می‌آید.

پس از دریافت یک قاب RGB-D (که شامل تصویر رنگی و عمق می‌باشد)، همانطور که گفته شد نقشه عمق محیط را به منظور کاهش نویز و بهبود کیفیت ردیابی دوربین، فیلتر می‌کنیم. به این دلیل که مقادیر عمق با عدم قطعیت بالا تاثیر منفی در کیفیت نقشه‌برداری دارند، آستانه را با توجه به فواصل گفته شده تعریف کرده و مقادیر عمق بالاتر از آستانه تعیین شده در نقشه عمق را صفر در نظر می‌گیریم. به منظور کاهش نویز و حفظ لبه‌ها ما نقشه عمق را با اعمال فیلتر bilateral فیلتر می‌کنیم(مانند [۲۰]). شکل ۴-۴ تاثیر گام‌های پیش پردازش برای نقشه عمق را نشان می‌دهد.



شکل (۴-۴): (الف) تصویر RGB و (ب) نقشه عمق از قاب RGB-D است. (پ) نقشه عمق به وسیله فیلتر bilateral پیش پردازش شده (ت) مقادیر عمق بیشتر از آستانه نادیده گرفته می‌شود.

۴-۴ ردیابی دوربین با استفاده از هم‌ترازی سه‌بعدی مبتنی بر ویژگی

یکی از مهم‌ترین قسمت‌های هر الگوریتم بازسازی سه‌بعدی مرحله ردیابی دوربین است، که برای هر قاب RGB-D بدست آمده موقعیت دوربین را تعیین می‌کند. معمولا، ردیابی دوربین به وسیله تخمین حرکت‌های نسبی دوربین از یک قاب به قاب دیگر اجرا می‌شود و در نهایت موقعیت جهانی دوربین می‌توانند بر مبنای این موقعیت‌های نسبی تعیین شوند، به این صورت که حالت مطلق اولین دوربین را (C_1) ماتریس $I \in \mathbb{R}^{4\times4}$ در نظر گرفته و برای بدست آوردن حالت‌های مطلق دوم به بعد آن‌ها را در تبدیل نسبی مرحله قبل ضرب می‌کنیم:

$$C_i = \mathfrak{T}(T_{i-1}^{i}.C_{i-1}) \qquad ۴-۳$$

روش‌های متفاوتی برای تعیین دقیق حالت سه‌بعدی نسبی بین قاب‌های متوالی وجود دارد:

- الگوریتم ICP [۲۵] دو ابر نقطه را بر مبنای یک حالت اولیه تخمین زده شده به وسیله حداقل کردن فاصله جفت نقاط دو ابر نقطه، هم‌تراز می‌کند.

- هم‌ترازی سه‌بعدی مبتنی بر ویژگی، ردیابی نسبتا دقیق دوربین[1] را بر مبنای ویژگی‌های بصری پراکنده[2] اجرا می‌کند. نتیجه تطبیق ویژگی بین دو تصویر رنگی دوبعدی که با استفاده از عمقشان به سه بعدی تبدیل می‌شوند، تناظر سه بعدی آن‌ها است. این تناظرها برای محاسبه حالت سه بعدی بین قاب‌ها استفاده می شوند.

در سالهای اخیر، راه‌کارهای بصری چگال[3] مانند [۲۶] و [۲۷] بیشتر مورد بررسی قرار گرفته اند. بر خلاف هم‌ترازی سه‌بعدی مبتنی بر ویژگی، که خیلی از اطلاعات درباره محیط را نادیده می‌گیرد، در این روش‌ها تمام اطلاعات رنگی و سه بعدی موجود استفاده می‌شود.

به منظور انتخاب راه‌کار مناسب برای این فرآیند، داده‌های RGB-D بدست آمده در سناریوی بازسازی محیط داخلی مورد نظرمان را بررسی می‌کنیم. محیط داخلی که می‌خواهد بازسازی شود معمولا ویژگی‌های رنگی متمایزی را بر روی سطح به نمایش می‌گذارد، در حالی که داده سه بعدی بدون اطلاعات رنگ، معمولا خیلی معنی دار نیست. از این رو، الگوریتم ICP برای ما راه‌کار مناسبی نیست، چرا که از اطلاعات رنگی به هیچ وجه استفاده نکرده و ممکن است زمانی که فقط با اطلاعات سه بعدی موجود کار می‌کنیم با خطا مواجه شود.

روش انتخاب شده این مزیت را دارد که هیچ مقدار دهی اولیه‌ای برای تعیین هم‌ترازی سه‌بعدی نیاز ندارد. اگرچه ما ردیابی دوربین را فقط برمبنای ویژگی‌های پراکنده اجرا می‌کنیم، ولی مدل‌های سه بعدی چگال می‌توانند با استفاده از تمام ابر نقطه‌هایی که حالت‌های آن‌ها تعیین شده است تشکیل شوند. با داشتن دو قاب RGB-D که همانطور در قسمت ۳-۴ گفته شد، هم‌ترازی سه‌بعدی دو به دو می‌تواند به چندین بخش تقسیم شود:

۱) نقاط کلیدی از دو تصویر RGB با جستجوی مکان های قابل تشخیص در تصاویر بر اساس ظاهر محلی آنها استخراج می شوند.

[1] Efficient camera tracking
[2] Sparse
[3] Dense visual

۲) توصیفگر فشرده به منظور نمایش این نقاط کلیدی مجزا از ناحیه دور آن‌ها استخراج می‌شود.

۳) ویژگی‌های شناسایی شده در دو تصویر به منظور یافتن تناظرهای نقاط ویژگی با هم منطبق می‌شوند.

۴) برای اینکه بتوانیم حالت نسبی سه‌بعدی بین قاب‌ها را محاسبه کنیم به ویژگی‌های دوبعدی متناظر مقادیر عمق اضافه می‌شوند. RANSAC به منظور حذف تطبیق‌های اشتباه در تناظرهای ویژگی استفاده می‌شود.

در ادامه ما قسمت‌های مختلف روش هم‌ترازی سه بعدی را با جزئیات بیان می‌کنیم.

۴-۴-۱ شناسایی ویژگی

در بینایی ماشین، مفهوم نقاط سودمند[1] که همچنین نقاط کلیدی یا نقاط ویژگی نامیده می‌شوند به طور گسترده‌ای برای حل تعداد زیادی از مسائل مانند بازشناسی شی، ردیابی بصری، بازسازی سه بعدی و ... استفاده می شود. این مفهوم بر این ایده متکی است که به جای آنکه کل تصویر را تحلیل کنیم، بهتر است بعضی از نقاط خاص را در تصویر انتخاب کرده و یک تحلیل محلی بر روی آن‌ها انجام دهیم. این روش مادامی که تعداد نقاط کافی در تصویر مورد نظر وجود داشته باشد و این نقاط دارای ویژگی‌های متمایز و پایدار باشند می‌توانند برای تخمین حالت دوربین مورد استفاده قرار بگیرند.

از آنجا که که این نقاط برای تحلیل محتوای تصویر استفاده می‌شوند، نقاط ویژگی باید به طور ایده‌ال در یک صحنه مشابه مجددا قابل پیدا شدن بوده، بدون توجه به اینکه از کدام نما، اندازه یا جهت تصویر گرفته شده است. حساس نبودن به تغییر نما یک ویژگی مطلوب در تحلیل تصویر بوده و موضوع مطالعات بسیاری است. همانطور که خواهیم دید، آشکارسازها[2] خصوصیات تغییرناپذیر متفاوتی دارند.

[1] Interest point
[2] Detectors

۴-۴-۱-۱ آشکارسازی گوشه‌ها در تصویر

هنگامی که برای نقاط ویژگی در تصـویر جسـتجو می‌کنیم، گوشـه‌ها به عنوان یک گزینه مطلوب هستند. آن‌ها در واقع ویژگی‌های محلی هستند که به راحتی می‌توانند در یک تصویر شناسایی شوند، و به علاوه، این ویژگی‌ها در صـحنه‌های سـاخت بشـر به فراوانی یافت می‌شـوند (در جایی که این ویژگی‌ها به وسیله دیوارها، درها، پنجره‌ها، میزها و ... می‌توانند تولید شوند). گوشه‌ها به این خاطر که ویژگی‌های دوبعدی هستند و می‌توانند به صورت دقیق یافت شوند مطلوب هستند.

به منظور اجرای هم‌ترازی سـه‌بعدی مبتنی بر ویژگی که در بخش ۴-۴ گفته شـد، مهم اسـت که بطور دقیق ویژگی‌های بصـری را پیدا کنیم و آن‌ها را به عنوان اولین گام به کار ببریم. بنابراین، مکان‌های متمایز در تصـویرهای رنگی باید طوری پیدا شـوند تا نقاطی را در تصـویرها فراهم کنند که بتوانند در دیگر تصـویرها نیز شـناسـایی شـوند. برای این منظور، روش‌هایی مانند گوشـه یاب هریس می‌توانند استفاده شوند [۲۸]. آشکارساز نقطه کلیدی بر روی تصویرهای خاکستری عمل کرده، که این تصویر خاکستری از تصویرهای رنگی تولید شده است.

زمانی که نقاط کلیدی پیدا شـدند، باید توصـیفگرهای مقاومی را که ناحیه محلی دور این نقاط کلیدی را توصیف می‌کنند استخراج کنیم. از آنجا که ناحیه‌های محلی حول نقطه کلیدی ممکن است از یک تصـویر به دیگری متفاوت باشـد، توصـیفگر باید در برابر تغییر اندازه، انتقال، چرخش و ... تغییر ناپذیر باشد. در عین حال، متمایز بودنشان بین تکه‌های مختلف نیز باید تضمین شود. برای این منظور، انواع مختلفی از توصیفگرها توسعه داده شده است و در دهه‌های قبل به طور موفقیت آمیزی استفاده شده که در ادامه به معرفی آنها می پردازیم:

SIFT[29][1]: احتمالا محبوب‌ترین و موفق‌ترین نوع توصیفگر آشکارساز ویژگی[2] است که توسط دیوید لویی به منظور فراهم کردن ویژگی‌های مقاوم و قابل تکرار برای انجام کارهایی مانند ردیابی و

[1] Scale Invariant Feature Transform
[2] Feature detector

بازشناسی توسعه داده شده است. این آشکارساز ویژگی به تغییرات اندازه و چرخش و البته تا حدودی به تغییرات رو شنایی حساس نیست. با این وجود،SIFT از نظر محاسباتی هزینه‌بر بوده و عموما برای کارهای آنی[1] مناسب نیست.

برای جبران این موضوع، نسخه مبتنی بر پردازنده گرافیکی آن که SiftGPU[30] نام دارد به منظور سرعت بخشیدن به محاسبه ویژگی‌های SIFT، توسعه داده شده است.

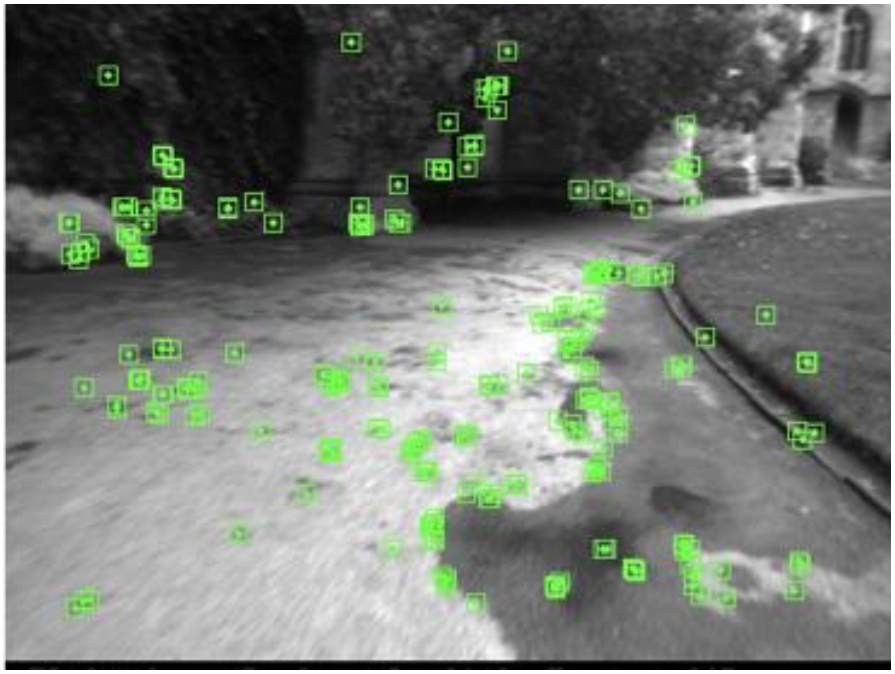
آشکارساز ویژگی SURF[2] [۳۱] که از SIFT الهام گرفته شده است ولی در مقایسه با SIFT هزینه محاسباتی کمتری داشته و سریع‌تر است ولی استفاده از آن در کاربردهای آنی هنوز محدود است.

با ORB[32][3] یک جایگزین سریعتر نسبت به موارد قبلی معرفی شده است. این آشکارساز برای این طراحی شده که هم نتایجی مانند SIFT و SURF داشته باشد و هم از سرعت خوبی برخوردار باشد.

تمام آشکارسازهای ویژگی که در بالا گفته شد به جز SiftGPU در روندنمای توسعه داده شده در این پایان‌نامه استفاده شده است. این کار از یک طرف به ما این اجازه را می‌دهد که برنامه تا حد ممکن منعطف باشد و در صورت نیاز می‌توانیم پایداری روش‌های متفاوت را برای کارمان ارزیابی کنیم.

در عمل، باید پارامترهای هر نوع آشکارساز ویژگی را طوری تعیین کنیم که تعداد نقاط کلیدی معقول باشند. تعداد ویژگی‌های زیاد، فرایند تطبیق را کند کرده و معمولا تعداد زیادی نقطه کلیدی اشتباه در فرایند تطبیق تولید می‌کنند. اگر تعداد نقاط کلیدی شناسایی شده خیلی کم باشد باعث می‌شود که هم‌ترازی سه بعدی قاب به علت کم بودن نقاط کلیدی متناظر با شکست مواجه شود. در شکل ۴-۵ نقاط کلیدی بدست آمده با استفاده از ORB را مشاهده می‌کنید.

[1] Real-time
[2] Speeded-Up Robust Features
[3] Oriented FAST and Rotated BRIEF



شکل (۴-۵): مثالی از نقاط کلیدی بدست آمده توسط Orb

هر نقطه کلیدی $(u_k. v_k)$در تصویر i به وسیله یک بردار توصیفگر d_k^i توصیف می‌شود. این بردار شامل تعداد معینی المان است(در حالت SIFT، ۱۲۸ المان داریم) که می‌تواند با بردارهای توصیفگرِ دیگر نقاط کلیدی در فرایند تطبیق مقایسه شود. به عنوان نتیجه نهایی از فرآیند آشکارساز ویژگی، ما یک مجموعه توصیفگر ویژگی $D_i = \{d_k^i\}$ از تصویر رنگی قاب i ورودی بدست می‌آوریم.

۴-۴-۲ تطبیق ویژگی

بعد از آنکه توصیفگرهای ویژگی شناسایی شدند می‌توانند برای ساختن تناظر بین دو تصویر j و i استفاده شوند. فرض می‌کنیم، تصویرها به اندازه کافی با هم هم‌پوشانی داشته تا بتوانیم هم‌ترازی سه‌بعدی را انجام دهیم. با این فرض، این مورد را در نظر می‌گیریم که ویژگی‌ها در یک تصویر تناظرهایی در تصویر دیگر دارند، اگرچه ممکن است بعضی از آنها به هر دلیلی دیده نشده یا در اثر انسداد پیدا نشوند.

به منظور پیدا کردن تطبیق‌ها برای یک بردار توصیفگر d_k^i در یک تصویر مانند i، باید آن را با توصیفگر d_l^j در تصویر دوم j مقایسه کنیم. به این خاطر که تشابه بین دو بردار قابل اندازه‌گیری باشد،

باید یک فاصـــله واقعی را اســـتفاده کنیم. در مورد ویژگی‌های SIFT و SURF، بردارهای توصـــیفگر می‌توانند با ا ستفاده از نرم اقلید سی($l2 - norm$) مقایسه شوند. اگر دو بردار تو صیفگر d_k^i وd_l^j n-بعدی داشته باشیم، نرم اقلیدسی به صورت زیر محاسبه می‌شود:

$$d(d_k^i . d_l^j) = \|d_k^i - d_l^j\|_{L2} = \left(\sum_{s=1}^{n} \left(d_{k_s}^i - d_{l_s}^j\right)^2\right)^{1/2} \qquad ۴-۴$$

همناطور که گفته شـد n در مورد SIFT ۱۲۸ المان اسـت. به منظور کارایی، ویژگی‌های ORB به وسـیله توصـیفگر باینری وصـف می‌شـوند که برای تطبیق از فاصـله Hamming آن‌ها به جای نرم اقلیدسی استفاده می‌شود. در مجموع، هر چه فاصله $d(d_k^i . d_l^j)$ دو بردار توصیفگر کمتر باشد احتمال اینکه این دو توصیفگر یک ویژگی را وصف کنند زیادتر است.

با فاصـله واقعی تعریف شـده در بالا، این امکان هسـت تا برای پیدا کردن تناظرها در تمام نقاط ویژگی دو تصویر متفاوت جستجو کنیم. با استفاده از یک استراتژی ساده مانند $brute - force$، هر تو صیفگر d_k^i تصویر i را با تمام تو صیفگرهای تصویر j مقایسه می‌کنیم و نزدیکترین تو صیفگر d_l^jرا تعیین می‌کنیم. شکل ۴-۶ که مرحله تطبیق ویژگی را نشان می‌دهد.

شکل (۴-۶): ویژگی‌های متناظر بین دو تصویر بعد از تطبیق ویژگی(که شامل تطبیق‌های اشتباه هم هست)[۳۳]

با این وجود، راه کار brute-force پیچیدگی در جه دو دارد $O(n * m)$ که m و n تعداد بردارهای توصیفگر در هر تصویر می‌باشند. برای تصویرهایی با تعداد زیادی از بردارهای ویژگی، فرایند تطبیق می‌تواند با استفاده از ساختارهای اندیس گذاری خاص مانند جستجو درخت‌های چند بعدی سریع‌تر اجرا شود. مانند مرحله شناسایی ویژگی، ما دوباره از OpenCV [۱۷] برای تطبیق ویژگی استفاده می‌کنیم. این کتابخانه برای ما تطبیق‌کننده توصیفگر $\text{brute} - \text{force}$ و همچنین یک تطبق‌گر بر مبنای جستجو نزدیکترین همسایه که بر اساس FLAAN است را فراهم می‌کند[۳۴].

بعد از مرحله تطبیق، ما جفت توصیفگرهای ویژگی متناظر $\{d_s^i\}$و $\{d_s^j\}$را بدست می‌آوریم. در اینجا، توصیفگرهای متناظر در دو تصویر j و i در موقعیت مشابه s در مجموعه متناظرشان هستند. با جستجو نقطه کلیدی $x_s = (u_s. v_s)^T \in \mathbb{R}^2$ مربوطه برای هر توصیفگر $\{d_s\}$ در تصویرها، ما جفت‌های مربوط به مجموعه‌های نقطه ویژگی دوبعدی را بدست می‌آوریم:

$$K = \{x_s^i\} \quad and \quad K = \{x_s^j\} \qquad ۵-۴$$

به منظور محدود کردن نیازهای محاسباتی، فقط ۵۰۰ تطبیق که کمترین فاصله را دارند برای پردازش‌های بعدی استفاده می‌شود.

۴-۵ هم‌ترازی با استفاده از RANSAC

مجموعه‌های متناظر نقاط ویژگی دوبعدی $K_j = \{x_s^j\}$ و $K_i = \{x_s^i\}$ که از مرحله تطبیق ویژگی بدست آمده‌اند می‌توانند برای تخمین حالت نسبی $T_i^j \in SE(3)$ بین حالت‌های دوربین استفاده شوند.

مادامی که ما برای هر نقطه کلیدی $x_s = (u_s. v_s)^T$ عمق $d_s = I_d(u_s. v_s)$ را داشته باشیم، طبق معادله ۱۴-۳ می‌توانیم مختصات سه بعدی نقطه کلیدی دوبعدی را بدست بیاوریم، به طوری که $p_s = \rho(u_s. v_s. d_s) \in \mathbb{R}^3$ است. که در نتیجه دو مجموعه نقطه سه بعدی P_j و P_i بدست می‌آید:

$$P_i = \{\mathbf{p}_s^i\} \quad \text{and} \quad P_j = \{\mathbf{p}_s^j\} \qquad ۶-۴$$

در تئوری، مستقیما می‌توانیم حالت سه بعدی نسبی بین دو قاب را با استفاده از هم‌ترازی مبتنی بر SVD که در فصل ۳ گفته شد با این نقاط سه بعدی متناظر بدست بیاوریم.

با این وجود، تطبیق‌های شاخص[1] بدست آمده در مرحله تطبیق ویژگی هنوز شامل خطا می‌باشند. مقدار عمق اشتباه یا صفر می‌تواند بر روی کیفیت هم‌ترازی محاسبه شده تاثیر بگذارد. در مورد سنسورهای RGB-D استفاده شده، نویز در مقادیر عمق باعث ایجاد خطا می شود، زیرا از دست دادن همزمانی بین دوربین رنگی و مادون قرمز باعث به وجود آمدن ناهماهنگی بین مقدار عمق تصویر با توجه به تصویر رنگی می‌شود.

برای بدست آوردن تناظرهای دقیق باید بین تطبیق‌های درست و مقادیر خطا تفاوت قائل شویم. این کار می‌تواند به وسیله جایگزین کردن تخمین تبدیل سه بعدی فصل ۳ در راه‌کار RANSAC صورت گیرد. استفاده از RANSAC به ما اجازه می‌دهد تا تخمین حالت نسبی را در برابر تطبیق‌های اشتباه مقاوم سازیم. RANSAC یک الگوریتم تکرار شونده بوده که به خوبی در مسائل بینایی ماشین شناخته شده است و به منظور تخمین پارامترهای یک مجموعه استفاده می‌گردد (شکل ۴-۷). برای اولین بار ۱۹۸۱ توسط Bolles و Fischler معرفی شد[۳۵]. برای هر تکرار، تعداد کمی از نمونه‌ها به منظور تعریف مدل به صورت تصادفی انتخاب می شوند. سپس این مدل برای تمام مجموعه داده به وسیله یک تابع خطا ارزیابی می‌شود. این فرآیند چندین بار با انتخاب نمونه‌های جدید تکرار شده و بهترین تبدیل پیدا شده را نگاه می‌دارد.(الگوریتم ۴-۱)

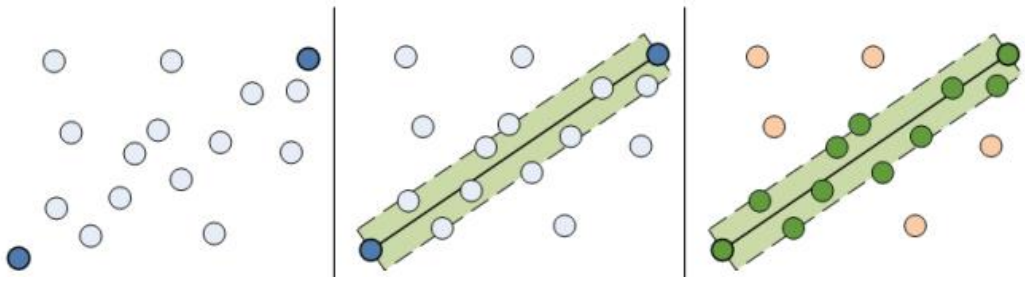
[1] Putative

Algorithm 1 General RANSAC

Require: Dataset of points

```
bestModel, bestInliers ← ∅
Define the number of iterations N
for iteration = 1 to N do
  samples ← Pickup k points randomly
  Compute currentModel from samples (base hypothesis)
  inliers ← ∅
  for all points do
    Evaluate currentModel for the point and compute its error
    if error < threshold then
      inliers ← inliers + point
    end if
  end for
  Count number of inliers and compute mean error
  if currentModel is valid then
    Recompute currentModel from inliers
    if currentModel better than bestModel then
      bestModel ← currentModel
      bestInliers ← inliers
    end if
  end if
end for
return bestModel, bestInliers
```

الگوریتم ۴-۱: الگوریتم کلی RANSAC[۳۶]



شکل(۴-۷): اجرای فرآیند RANSAC برای مجموعه داده دوبعدی. (سمت چپ) ابتدا z نمونه به صورت تصادفی انتخاب می‌شوند.(که z = 2 است). سپس(وسط) با استفاده از دو نقطه انتخاب شده مدلی را تعریف می‌کنیم. در اینجا مدل ما یک خط می‌باشد. سپس آستانه‌ای را برای ارزیابی نقاط درست و اشتباه تعریف می‌کنیم. (سمت راست) به منظور ارزیابی مدل فاصله هر نقطه تا مدل تعریف شده را بدست می‌آوریم. این عمل باعث می‌شود که نقاط ما به دو دسته نقاط درست(که با سبز مشخص شده) و نقاط نادرست تقسیم شوند. نسبت تعداد نقاط درست به کل نقاط، امتیاز مدل مورد نظر را مشخص می‌کند. هر چه مقدار این امتیاز بیشتر باشد مدل بهتر است.

چالش اصـــلی الگوریتم RANSAC تعیین تعداد تکرار و آســـتانه‌ها می‌باشـــد. فرض کنید مجموعه تطبیق از $W\%$ تطبیق در ست تشکیل شده با شد، سپس احتمال اینکه ما k تطبیق در ست انتخاب کنیم برابر است با W^K. در نهایت احتمال اینکه یک مجموعه شامل حداقل یک تطبیق نادرست باشد برابر با $(1 - W^K)$ می‌باشـــد. اگر N دفعه این فرآیند را انجام دهیم، احتمال اینکه یک مجموعه تصادفی که فقط شامل تطبیق‌های درست باشد برابر است با $p = 1 - (1 - W^K)^N$. در نهایت تعداد تکرار الگوریتم به صورت زیر محاسبه می‌شود:

$$N = \frac{\log(1-p)}{\log(1-W^K)} \qquad ۷-۴$$

تعیین مقدار مناسب برای آستانه به نوع مسئله و برنامه بستگی دارد ولی در حالت کلی این مقدار به صورت تجربی تعیین می‌گردد. بعد از آنکه الگوریتم RANSAC تمام شد، ما تبدیل T_j^iرا با بزرگترین مجموعه وفاق[1] بدست می‌آوریم.

بعد از آنکه ما تناظرهای نقاط سه بعدی بدون خطا را بد ست آوردیم آن‌ها را درمجموعه‌های $\hat{P}_i$ و $\hat{P}_j$ قرار می‌دهیم. فقط از تناظرهای در ست برای هم‌ترازی سه‌بعدی در مراحل بعدی پردازش ا ستفاده می‌کنیم. اگر تعداد نمونه‌های منطبق شده از آستانه تعیین شده توسط کاربر کمتر بود، هم‌ترازی سه بعدی با شکست مواجه می‌شود. این اتفاق به این خاطر رخ می‌دهد که دو قاب که صحنه‌های متفاوت را نشـان می‌دهند هم‌پوشـانی کافی از یک صـحنه یکسـان ندارند. شـکل ۴-۸ فرآیند تطبیق را بعد از اعمال RANSAC بر روی مجموعه داده P_j و P_i را نشان می‌دهد.

[1] Consensus

شکل(۴-۸): ویژگی‌های متناظر بعد از حذف تطبیق‌های اشتباه توسط روش RANSAC[33]

درحالی که RANSAC یک راه‌کار کلی است، PROSAC برای تطبیق ویژگی مقاوم خیلی سریع‌تر است[۳۷].

۴-۶ اندازه‌گیری حرکت دوربین

برای اینکه ماتریس تبدیل بدست آمده بین دو قاب دقیق باشد، بهتر است که میزان جابه‌جایی دوربین را اندازه‌گیری کنیم. به این منظور بعد از آنکه تبدیل نسبی بین دو قاب متوالی را بدست آوردیم، با استفاده از نُرم دو بردار چرخش و انتقال مقداری را بدست آورده و با میزان آستانه تعیین شده مقایسه می‌کنیم و اگر این مقدار بدست آمده از آستانه مشخص شده بیشتر بود، این قاب را نادیده گرفته و مجددا قاب جدیدی را دریافت می‌کنیم. مقدار این آستانه با سعی و خطا بدست می‌آید.

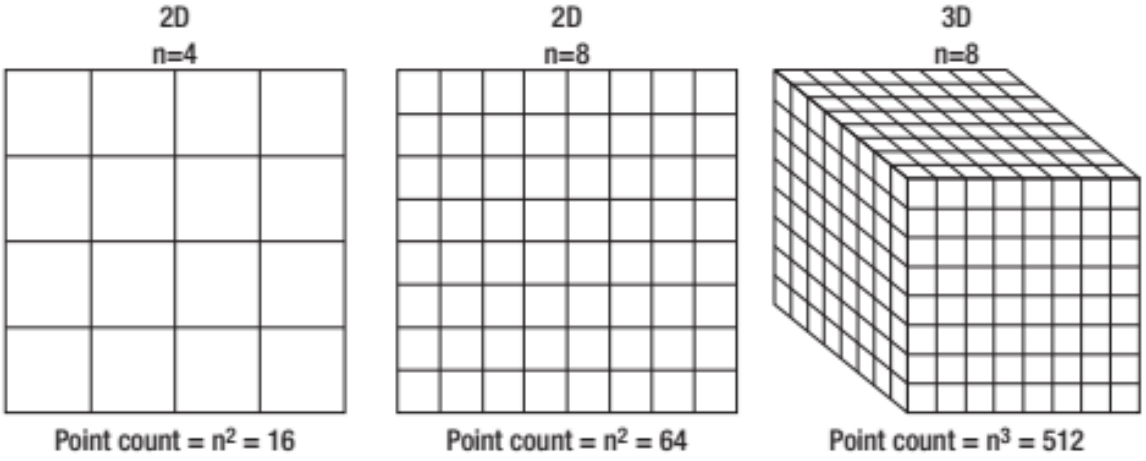
۴-۷ بررسی ردیابی دوربین

پس از اندازه‌گیری حرکت دوربین بررسی می‌کنیم که ردیابی دوربین از دست نرفته باشد. وقتی که یک قاب به علت سریع بودن حرکت دوربین نادیده گرفته شود و بعد از آن ما قاب جدیدی را ذخیره کنیم، این عمل باعث می‌شود که مقدار هم‌پوشانی دو تصویر کم باشد و در نتیجه نقاط کلیدی متناظر در دو تصویر به اندازه کافی وجود نداشته باشد. برای رفع این مشکل طبق روندنمای گفته شده

در شـــکل ۴-۱ به آرامی دوربین را به چند موقعیت قبل برگردانده و فرآیند شـــناســـایی نقاط ویژگی و هم‌ترازی سه‌بعدی را دوباره انجام می‌دهیم.

۴-۸ فرآیند تولید و ذخیره سازی ابر نقطه

قبل از اینکه فرآیند تولید ابر نقطه ســـه‌بعدی را با کینکت شـــروع کنیم، ابتدا باید بدانیم که این مدل‌ها را چطور ذخیره کرده و نمایش دهیم. اکثر توسعه دهنده‌ها با چگونگی مصور سازی تصویرهای دوبعدی و ذخیره کردن آن‌ها آ شنا ه ستند. ت صویر شامل یک شبکه متداول از پیک سل‌ها ا ست که فضـــای دوبعدی را نمونه برداری می‌کنند. دقت تصـــویر(به این معنی که چقدر فضـــای دوبعدی نمونه برداری شده چگال ا ست) نه تنها کیفیت ت صویر را تعیین می‌کند بلکه نیازهای پرداز شی و حافظه را نیز مشـــخص می‌کند. برای مثال همانطور که قبلا اشـــاره شـــد تصـــویر رنگی کینکت دارای دقت ۶۴۰×۴۸۰ اســت به این معنی که دارای ۳۰۷,۲۰۰ پیکســل اســت. اما اگر به عنوان مثال دقت را دو برابر کنیم (۱۰۲۴×۹۶۰) تعداد پیک سل‌ها ۴ برابر می شود(شکل ۴-۹). خو شبختانه رایانه‌های جدید می‌توانند تصـــویرهای چندین مگاپیکســـلی را بدون مشـــکل پردازش کنند. در عین حال، زمانی که با نمایش سه بعدی داده‌ها سر و کار داریم انتخاب‌های نمایشی زیادی وجود دارد که هر کدام از آن مزایا و معایب خود را دارند.

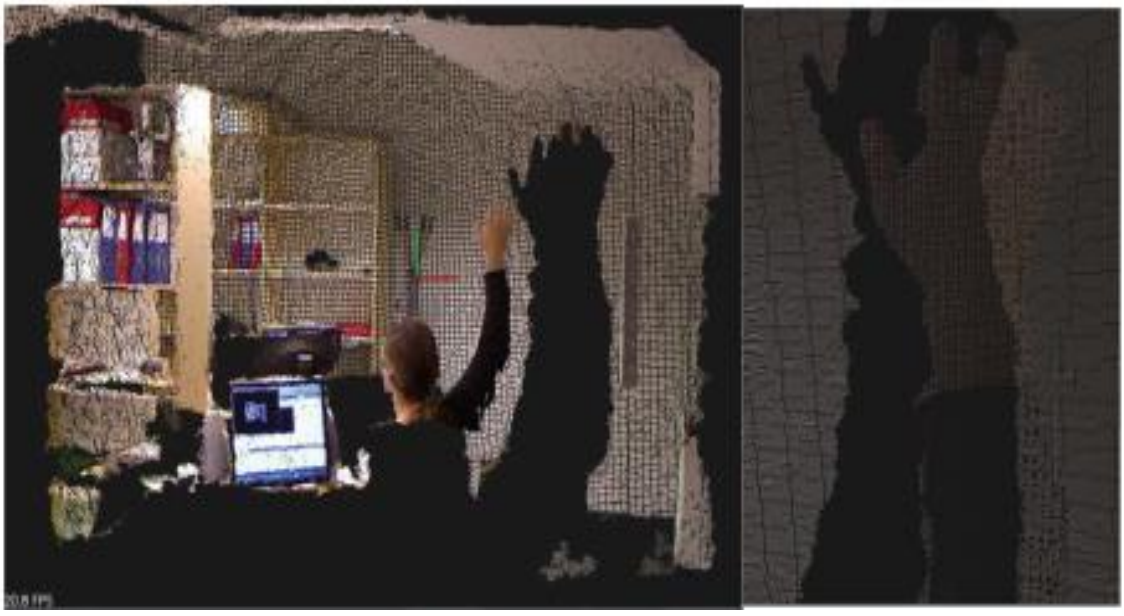


شکل (۴-۹): انواع ابر نقطه در فضای دوبعدی و سه‌بعدی

۴-۸-۱ ابرهای نقطه

ابر نقطه مجموعه‌ای از نقاط سه بعدی نامتصل است. ما آن‌ها را ابر می‌نامیم زیرا زمانی که این نقاط را تصویر می‌کنیم، نبود اتصال بین این نقاط، آن‌ها را اینگونه نشان می‌دهد که ظاهرا در فضا شناور هستند. ساده‌ترین نوع ابر نقطه فقط شامل اطلاعات موقعیت است، ولی هر نقطه می‌تواند شامل دیگر خصوصات(مانند رنگ و جهت نرمال) باشد. ابر نقطه می‌تواند به عنوان یک مدل جهانی استفاده شود. شکل ۴-۱۰ در سمت چپ تمام ابرنقطه دیده می‌شود و در سمت راست یک قسمت زوم شده را نشان داده شده است. اگرچه این ابر نقطه از دور یکپارچه و به هم پیوسته به نظر می‌رسد ولی وقتی بزرگنمایی کنیم عدم اتصال نقاط نمایان می‌شود.

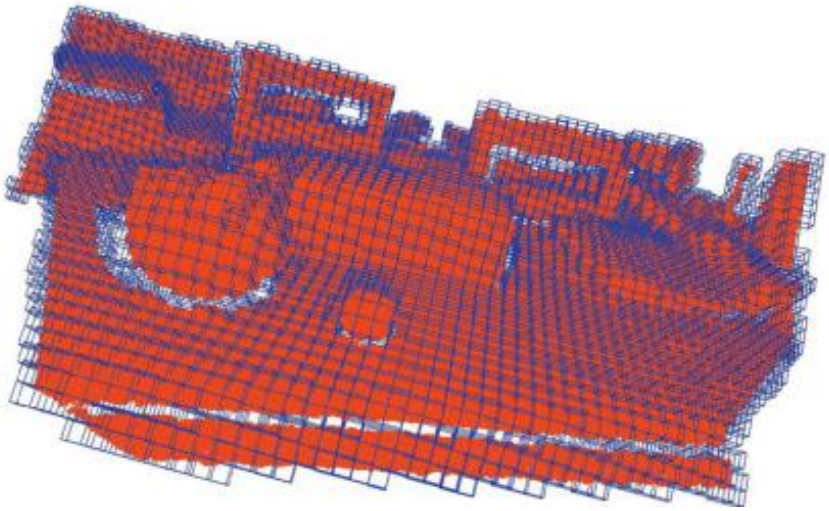
ابر نقطه، یک راه طبیعی برای نمایش اطلاعات بدست آمده از کینکت است، زیرا هر پیکسل دوبعدی در تصویر عمق همانطور که در رابطه ۳-۱۴ گفته شد می‌تواند به یک نقطه سه بعدی تبدیل شود. ما برای ذخیره ابر نقطه از کتابخانه ابر نقطه (PCL) استفاده می‌کنیم.



شکل (۴-۱۰): ابر نقطه ساخته شده توسط PCL

۴-۸-۲ واکسل‌ها

یک راه برای نمایش ابر نقطه سه بعدی این ا ست که یک شبکه سه‌بعدی منظم که ف ضای سه بعدی را نمونه برداری می‌کند بسازیم. این روش مدلی را می‌سازد که به عنوان تصویر حجمی شناخته می‌شود، و المان‌های این نوع شبکه واکسل‌ها بوده و مدل سه بعدی پیکسل‌ها می‌باشند(شکل ۴-۱۱). با این وجود، همانطور که در بخش قبل گفتیم زمانی که وارد دنیای سه بعدی می‌شویم، تعداد پیکسل‌ها به صورت نمایی رشد پیدا می‌کند. برای مثال اگر تصویر VGA را که دقت پایینی دارد حول بعد سومش گسترش دهیم(۶۴۰*۴۸۰×۴۸۰)، باید ۱۴۷,۴۵۶,۰۰۰ پیکسل داشته باشیم که معادل یک تصویر ۱۴۰ مگاپیکسل است.

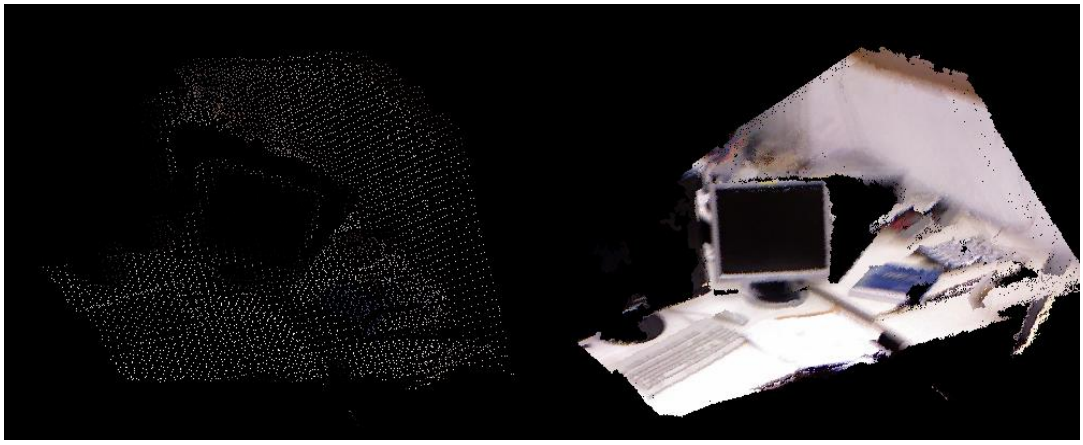


شکل (۴-۱۱): نمونه‌ای از ابر نقطه نمایش داده شده توسط واکسل

۴-۸-۲-۱ دلیل استفاده از واکسل

دلیل اصلی استفاده از واکسل این است که این کار اندازه مجموعه داده را ، تا حدود مناسبی کاهش می‌دهد و این روش برای کار بر روی داده‌های بزرگ زمانی که به صورت آنی پردازش می‌کنیم، ضروری است. برای مثال اگر محیطی سه متر در سه متر در سه متر داشته باشیم که شامل ۵۰۰,۰۰۰

نقطه با شد می‌توانیم با سایز واک سل ۵ سانتی‌متر تعداد نقاط را به بیش از ۵۰ در صد کاهش دهیم. (شکل۴-۱۲)



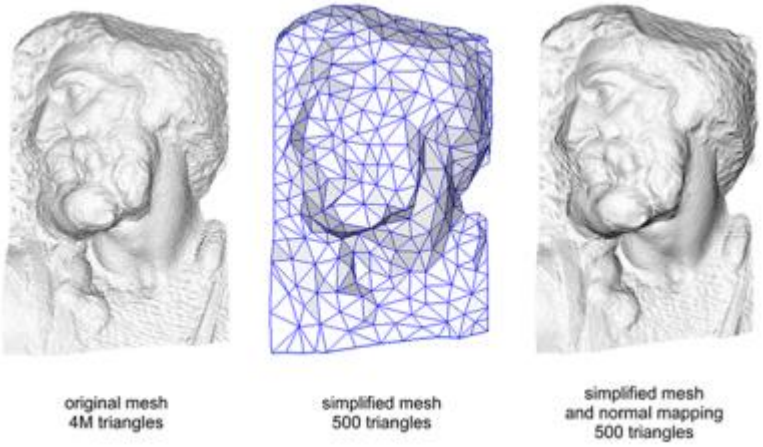
شکل(۴-۱۲): (سمت چپ) ابر نقطه ساخته شده با واکسل ۱۰ سانتی‌متر (سمت راست) همان ابر نقطه با واکسل ۱ سانتی‌متر

۴-۸-۳ مدل مش

مدل مش یک راه خیلی فشـرده برای نمایش داده‌های سـه بعدی اسـت و این روش برای نمایش انیمیشن‌های سه بعدی و بازی‌های رایانه‌ای انتخاب می شود. آن‌ها سطح یک جسم را با استفاده از چند ضلعی‌ها (معمولا مثلث‌ها) مدل می‌کنند. نواحی تخت[1] چند ضلعی‌های کمتری نیاز دارند، درحالی که نواحی پیچیده می‌تواند برای نمایش دادن سطحی با جزییات بالا تعداد زیادی از این چند ضلعی‌ها را استفاده کنند.

مدل‌های مش برای نمایش داده‌های سـه بعدی خیلی منعطف و کارا اسـت، ولی سـاختن آن‌ها سخت است. توسعه‌دهندگان سه بعدی موقعیت رئوس و جهت نرمال را انتخاب کرده و اتصال سطح را تعیین می‌کنند. با این وجود، ما از کینکت موقعیت سه بعدی و رنگ را می‌گیریم و هنوز هیچ اطلاعاتی درباره جهت سطح یا اتصال پیکسل‌ها در فضای سه بعدی نداریم که البته ساخت مدل مش خارج از حوزه این پایان نامه است.(شکل ۴-۱۳)

[1] Flat area



شکل (۴-۱۳): نمونه‌ای از فرآیند مش زنی:(سمت چپ) مدل ساخته شده با ۴ میلیون مش مثلثی،(وسط) مدل ساخته شده با ۵۰۰ مش مثلثی،(سمت راست) مدل ساخته شده بوسیله ۵۰۰ مش و نرمال

۴-۸-۴ ساخت ابر نقطه با استفاده از PCL

نوع پایه‌ای این کتابخانه pcl::PointCloud<PointT> اســـت. این نوع پایه‌ای به تنهایی چیزی بیشـــتر از یک بردار نقاط اســـت و این کتابخانه امکانات بیشـــتری مانند مصـــورســـازی، تخمین نرمال و ورودی/خروجی را برای ما فراهم می‌کند.

کلاس PointCloud امکان کار کردن با انواع نقاطی که ما می‌خواهیم ذخیره کنیم دارد. ساده‌ترین نوع داده ابر نقطه pcl::PointXYZ است، که فقط شامل اطلاعات سه بعدی موقعیت است.

نوع دیگر، pcl::PointXYZRGB ا ست که اطلاعات رنگ را نیز برای هر نقطه ذخیره می‌کند. این کتابخانه انواع مختلف دیگری را نیز که برای نیازهای رایج مورد نیاز اســـت، فراهم می‌کند. نوع داده دلخواه هم می توان تعریف کرد. اگر تعداد نقاط از قبل معلوم باشـــد می‌توانیم به ابر نقطه حافظه اخـتصـــاص دهـیـم، در غـیـر ایـنصـــورت هـر نـقـطـه مـی‌تـوانـد بـه وســـیلـه تـابـع PointCloud::push_back(PointT) به ابر نقطه ا ضافه شود. متد push_back به سادگی هر نقطه را به تنهایی به ابر ا ضافه کرده در حالی که حافظه را هم به صورت پویا[1] اختصاص می‌دهد. البته این کار باعث کاهش سرعت پردازش می شود. به این خاطر که برای د ستگاه‌های اندازه‌گیری سه بعدی

[1] Dynamically

مانند کینکت متداول است تا نقاط سه بعدی را به صورت یک شبکه دوبعدی برگرداند، ابرهای نقطه می‌توانند به صورت خطی و نیز دو بعدی اندیس دهی شوند.

۴-۸-۴-۱ رنگ کردن ابر نقطه

ابر نقطه‌ای که در مرحله قبل ساخته شد فقط شامل اطلاعات سه بعدی موقعیت بود. به این خاطر که کینکت علاوه بر عمق یک تصویر رنگی را نیز برای ما فراهم می‌کند، ضروری است که اطلاعات رنگ برای هر عمق به ابر نقطه اضافه شود. برای این کار، نقاطی را که از تصویر عمق بدست آورده‌ایم باید به دستگاه مختصات دوربین رنگی انتقال دهیم. چون دوربین‌های RGB و عمق، فاصله کانونی متفاوت و مبداهای مختلفی دارند، این کار نمی‌تواند در حالت دوبعدی انجام شود. برای هم‌تراز کردن تصویر رنگی و عمق از روش بیان شده در فصل ۳ استفاده می‌کنیم.

۴-۸-۴-۲ مصور سازی ابر نقطه

پس از ساختن ابر نقطه، برای اینکه بفهمیم روند را درست انجام داده‌ایم، باید آن‌ها را نمایش دهیم. دو راه رایج برای نمایش ابر نقطه وجود دارد: از کتابخانه PCL استفاده کنیم یا نقاط را خودمان به وسیله OpenGL به رسم کنیم. اگرچه PCL کار را برای ما ساده می‌کند ولی همواره بهترین گزینه نیست.

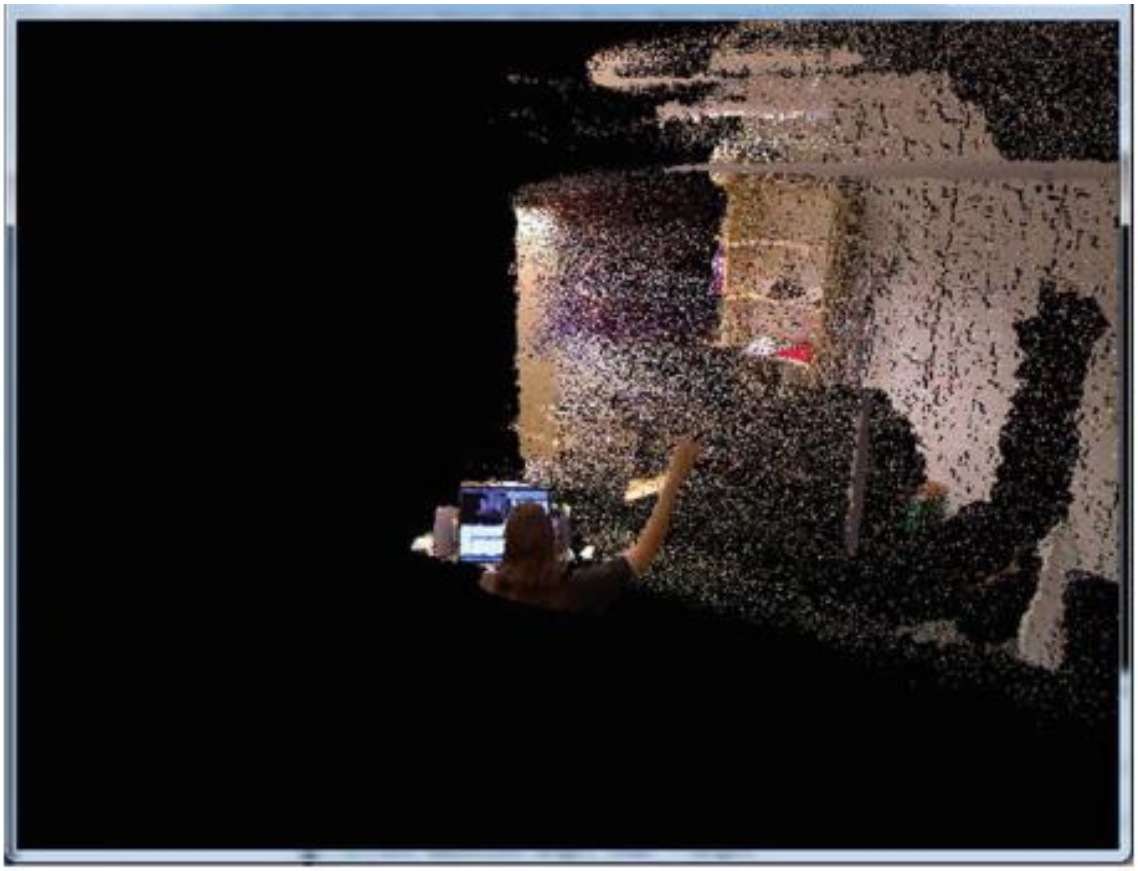
۴-۸-۴-۳ نمایش به وسیله PCL

مصور سازی به وسیله PCL نیازمند کتابخانه VTK است که می‌تواند از سایت www.vtk.org دانلود شود. البته اگر VTK[1] را نداشته باشید PCL همچنان کار می‌کند ولی برخی از قابلیت‌های آن غیرفعال می‌شود. ساده‌ترین راه برای نمایش ابر نقطه استفاده از pcl::visualization::CloudViewer است.

[1] Visualization toolkit

۴-۴-۸-۴ نمایش به وسیله OpenGL

بسته به نوع پروژه‌ای که می‌خواهیم بسازیم، ممکن است نیاز باشد فرآیند مصورسازی[1] را خودمان انجام دهیم. برای ا ستفاده از OpenGL از کتابخانه GLFW ا ستفاده می‌کنیم که رندر کردن، مدیریت رویدادها و ساخت پنجره را برای ما انجام می‌دهد.(شکل ۴-۱۴)



شکل (۴-۱۴): نمایی از پنجره ساخته شده توسط OpenGL

۴-۹ نقشه‌برداری

سیستم‌های SLAM[2] معمولا نقشه سه‌بعدی داخلی محیط را نگهداری و به روز رسانی می‌کنند. ما از این نقشـــه ســـه بعدی برای ردیابی دوربین و فراهم کردن تخمین‌های اولیه حالت‌های دوربین در دستگاه مختصات جهانی استفاده می‌کنیم. در این قسمت فرایند ساخت نقشه سه‌بعدی توسط ابر نقاط

[1] visualization
[2] Simultaneous localization and mapping

ساخته شده در بخش پیشین گفته می‌شود، که به صورت پیوسته در حال بروز رسانی بوده و قاب‌های جدید به آن اضافه می‌شود.

۴-۱۰ بازسازی چگال سه‌بعدی مدل‌ها

یکبار که موقعیت‌های نسبی $T_i \in SE(3)$ برای تمام دو جفت قاب‌های ورودی تعیین و تصحیح شد، هر قاب RGB-D بدست آمده می‌تواند به ابر نقطه سه بعدی کلی ما اضافه شود. برای این منظور، ما ابتدا یک ابر نقطه چگال رنگی سه‌بعدی از هر قاب RGB-D به واسطه بازافکنش پیکسل‌های تصویر رنگی به روی دستگاه مختصات سه‌بعدی با استفاده از معادله ۳-۱۴ بر مبنای مقادیر عمقشان می‌سازیم. مدل کامل صحنه به وسیله ترکیب مجزا هر ابر نقطه به یک نمایش واحد که ترکیبی از اطلاعات تمام قاب‌ها است ساخته می‌شود. شکل‌های زیر مدل‌های بدست آمده بعد از فرآیند هستند. (شکل ۴-۱۵)

شکل(۴-۱۵): ابر نقطه بدست آمده از ۱۰ قاب متوالی با واکسل ۱ سانتی‌متری

فصل پنجم: ارزیابی و نتایج تجربی

۵-۱ مقدمه

در این فصل کارایی سیستم بازسازی سه بعدی خود را ارزیابی می‌کنیم و نتایج روش‌های گفته شده در فصل‌های قبل را در اینجا نشان می‌دهیم. ابتدا با استفاده از چند مجموعه داده سیستم را آزمایش کرده و سپس با استفاده از سنسور کینکت، خودمان بازسازی را انجام می‌دهیم. سیستم مورد استفاده ما یک لپ‌تاپ با سی پیو اینتل core i5 با ۸ گیگ رم و کارت گرافیک Geforce GT 540m است.

۵-۲ ارزیابی کارایی

برای اثبات کارایی سیستم، ما علاوه بر استفاده از داده‌های RGB-D که خودمان از کینکت بدست آورده ایم از مجموعه داده‌های متفاوت نیز استفاده می‌کنیم و کارایی سیستم را با آن‌ها نیز بررسی می‌کنیم. و همچنین انعطاف‌پذیری سیستم را نشان می‌دهیم که این سیستم علاوه بر استفاده در محیط‌های داخلی می‌تواند برای انواع داده‌های RGB-D از مجموعه داده‌های متفاوت نیز کاربرد داشته باشد و آن‌ها را می‌تواند به صورت سه‌بعدی بازسازی کند. با ارزیابی سیستم پیشنهادی با مجموعه داده‌های TUM، اطلاعات دقیقی از دقت سیستم بدست می‌آوریم.

۵-۲-۱ ارزیابی با مجموعه داده TUM

بنچمارک TUM که در [۳۸]گفته شده است یک بنچمارک کامل برای ارزیابی سیستم‌های SLAM که بر مبنای RGB-D هستند مناسب است. این مجموعه داده شامل تعداد زیادی از تصویرهای RGB-D است که در شرایط محیطی و تحت حرکت‌های متفاوت دوربین گرفته شده است. این تفاوت‌ها در این مجموعه داده می‌تواند نشان دهد که سیستم طراحی شده SLAM در شرایط متفاوت عمومیت دارد یا نه.

برای هر صحنه، تصاویر رنگی و عمق همراه با اطلاعات دقیق موقعیت دوربین که توسط سیستم موشن کپچر با دقت بالا گرفته شده است را داریم. در دسترس بودن داده‌های دقیق[1] به ما این اجازه را می‌دهد که بدون اینکه بخواهیم خودمان این اطلاعات دقیق را بدست بیاوریم(که کار مشکلی هم هست)، دقت روش SLAM پیشنهادی را بررسی کنیم.

با توجه به مزیت‌های گفته شده در بالا، بنچمارک TUM به عنوان مجموعه داده‌ای مناسب برای ارزیابی روش‌های باز سازی سه‌بعدی است. تمام این مجموعه داده‌ها، تصویر را در نرخ ۳۰ هرتز و در ابعاد ۴۸۰ × ۶۴۰ ضبط کرده‌اند.

۵-۲-۲ مجموعه داده‌ها

برای ارزیابی روش استفاده شده در این پایان‌نامه، ما تعدادی از قاب‌های فراهم شده در بنچمارک TUMرا انتخاب می‌کنیم(شکل ۵-۱).

۱) FR1/XYZ: شامل تصویرهایی از یک میز داخل اتاق است. این مجموعه شامل تصاویری است که فقط حرکت انتقالی در راستای محورهای اصلی کینکت دارند در حالی که جهت تقریبا ثابت نگاه داشته شده است.

۲) FR2/PIONEER_360: این تصاویر توسط دوربین کینکت که در بالای ربات نصب شده است گرفته شده‌اند.

۳) FR1/ROOM: این مجموعه شامل تصاویری است که محیط تمام یک دفتر را تصویر برداری کرده و شامل loop_closure است.

۴) FR1/360: مجموعه‌ای از تصاویر که شامل یک چرخش ۳۶۰ درجه در محیط دفتر است.

۵) FR2/RPY: مجموعه‌ای از تصویرها که فقط شامل حرکت چرخشی است(شکل ۵-۲).

[1] Ground truth

خصوصیات این مجموعه داده‌ها در جدول ۵-۱ آورده شده است.

جدول(۵-۱): برخی از مجموعه داده‌های TUM

نام مجموعه داده	مدت زمان(ثانیه)	مدت زمان با Ground-truth	طول مسیر(متر)	ابعاد مسیر
FR1/XYZ	۳۰,۰۹	۳۰,۰۰	۷,۱۱۲	0.46m x 0.70m x 0.44m
FR1/RPY	۲۷,۶۴	۲۷,۴۲	۱,۶۶۴	4.24m x 4.38m x 0.06m
FR1/ROOM	۴۸,۹۰	۴۸,۸۷	۱۵,۹۸۹	2.54m x 2.21m x 0.51m
FR1/360	۲۸,۶۹	۲۸,۷۰	۵,۸۱۸	0.54m x 0.46m x 0.47m
FR2/PIONEER_360	۷۲,۷۵	۷۲,۰۰	۱۶,۱۱۸	2.44m x 1.47m x 0.52m

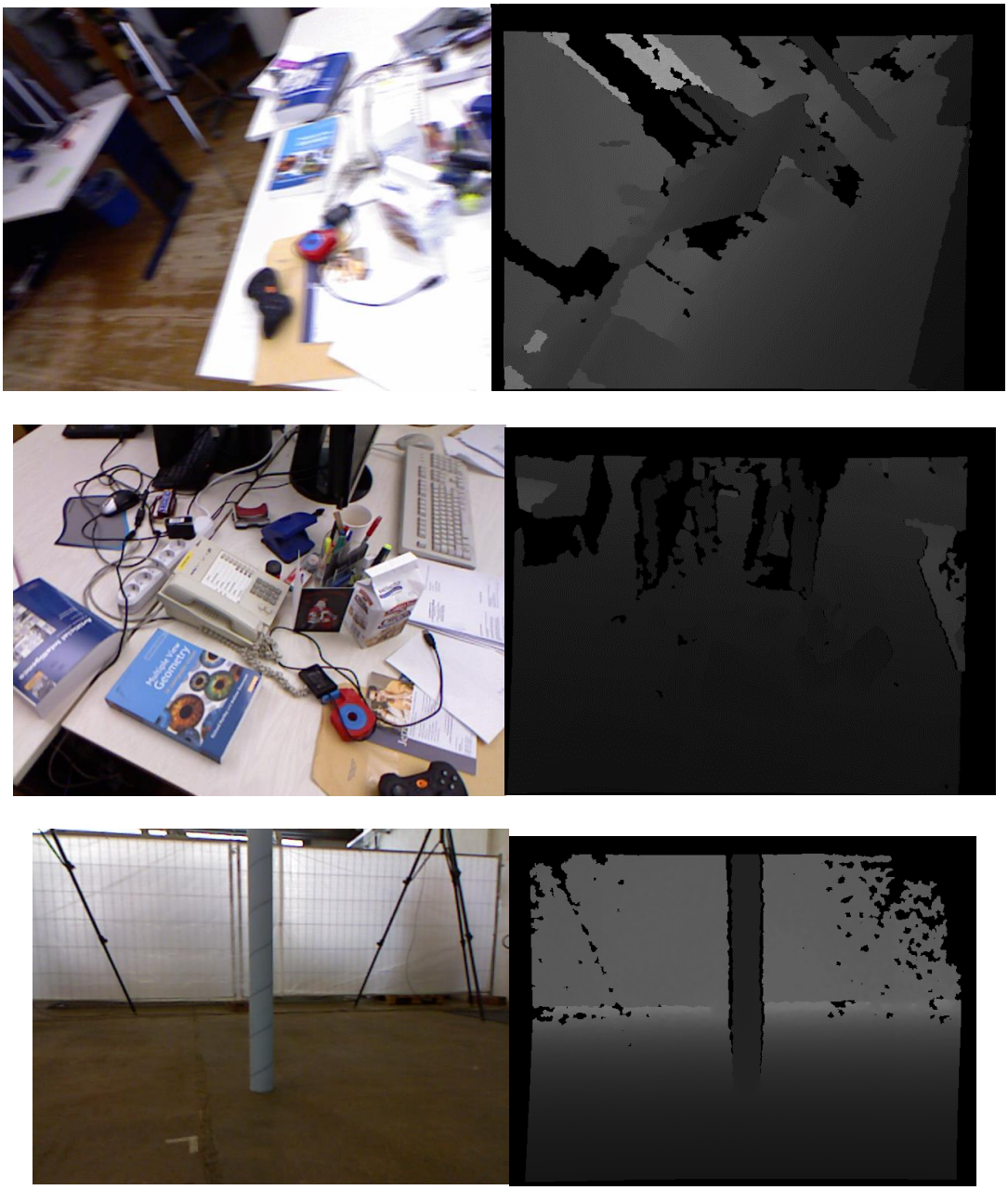
(الف) محیط داخلی دفتر

(ب) سالن مجموعه صنعتی

(پ) کینکت به همراه مارکر

(ت) کینکت نصب شده بر روی ربات

شکل (۵-۱): مجموعه داده‌های مختلف استفاده شده



شکل (۵-۲): تعدادی از تصاویر موجود در مجموعه داده‌های TUM(رنگی و عمق)

۵-۲-۳ ارزیابی واقعی

برای ارزیابی، فرض ما بر این ا ست که از توالی تصویرها مسیرهای تخمین زده شده (مقادیری که خودمان بدســـت آورده‌ایم) $P_1 \ldots . P_n \in SE(3)$ و مقادیر واقعی $Q_1 \ldots . Q_n \in SE(3)$ را داریم. برای ســاده ســازی، فرض می‌کنیم که توالی تصــویرها هم زمان بوده، هم اندازه نمونه برداری شــده و هر دو

آن‌ها طول n دارند. هر دو توالی شامل ماتریس‌های تبدیل همگن بوده که حالت تصویر RGB کینکت را در یک دستگاه مختصات (دلخواه) بیان می‌کنند.

در واقع انتخاب دستگاه مختصات بر روی کینکت اختیاری است، ما دوربین RGB را به این خاطر که تصاویر عمق را بر روی قاب‌های این دوربین هم‌تراز کرده‌ایم بعنوان مرجع استفاده می‌کنیم.

۵-۲-۳-۱ خطای مطلق مسیر(ATE[1])

برای سیستم‌های مبتنی بر SLAM بصری، پایداری سراسری مسیر حرکت یک کمیت مهم است. پایداری سراسری می‌تواند با مقایسه فواصل مطلق بین مسیر تخمین زده شده و مقادیر واقعی بدست بیاید. چون که هر دو مسیر در دستگاه‌های مختصات دلخواه هستند ابتدا نیاز است که آن‌ها را بر روی هم منطبق کنیم. این مسئله با استفاده از روش Horn[39] قابل دستیابی است که تبدیل S را طوری پیدا می‌کند که مسیر تخمین زده شده $P_{1:n}$ را بر روی مقادیر واقعی $Q_{1:n}$ نگاشت می‌کند. خطای مطلق مسیر در بازه زمانی i به صورت زیر محاسبه می‌شود:

$$F_i = {Q_i}^{-1} S P_i$$

RMSE در تمام بازه زمانی به صورت زیر محاسبه می‌شود

$$RMSE(F_{1:n}) \coloneqq \left(\frac{1}{n} \sum_{i=1}^{n} ||trans(F_i)||^2 \right)^{\frac{1}{2}}$$

که n تعداد حالت‌های دوربین و $trans(F_i)$ قسمت انتقالی ماتریس F_i است

۵-۲-۴ سیستم بازسازی سه‌بعدی مبتنی بر RGB-D

در این بخش ما کارایی سیستم و زمان اجرای آن را با استفاده از داده‌های بدست آمده تو سط خودمان و دیگر مجموعه داده‌ها را بررسی می‌کنیم. در نهایت هم نقشه سه‌بعدی بازسازی شده به همراه مسیر حرکت کینکت را نمایش می‌دهیم.

[1] Absolute trajectory error

ابتدا زمان پردازش و بازســـازی را برای داده های خودمان و ســـپس برای مجموعه داده ها را اندازه‌گیری می‌کنیم و ســـپس الگوریتم اســـتخراج ویژگی را تغییر داده و دوباره همین روند را تکرار می‌کنیم. ما برای ا ستخراج ویژگی همانطور که گفته شد از روش‌های SIFT، SURF و ORB ا ستفاده می‌کنیم. این الگوریتم‌ها از توابع کتابخانه OpenCV [10] هستند که از آن‌ها استفاده می‌کنیم.(جدول ۵-۲)

جدول (۵-۲): میانگین زمان اندازه‌گیری شده آشکارسازی و استخراج ویژگی(میلی‌ثانیه) برای الگوریتم‌های مختلف استخراج ویژگی

نام مجموعه داده	تعداد قاب‌ها	ORB	SIFT	SURF
FR1/XYZ	۶۲۰	**۱۶,۴**	۳۳۰,۰	۲۳۵,۰
FR1/RPY	۷۰۰	**۱۷,۶**	۲۵۱,۰	۱۹۵,۶
FR1/ROOM	۸۰۰	**۱۸,۱**	۳۰۰,۰	۲۳۳,۰
FR1/360	۳۶۰	**۱۳,۳**	۲۸۰,۰	۱۵۵,۰
FR2/PIONEER_360	۶۰۰	**۲۰,۰**	۲۲۶,۹	۱۸۹,۰
میانگین		**۱۷,۰**	۲۷۷,۵۸	۲۰۱,۵۲

با توجه به نتایج بدســـت آمده از جدول ۵-۲ می‌توان متوجه شـــد که ORB در حدود ۱۲ برابر ســـریع‌تر از دو الگوریتم SURF و SIFT می‌باشـــد. دو الگوریتم SIFT و SURF با اینکه نتایج نســـبتا مناســـبی دارند ولی برای پردازش منابع بیشـــتری اســـتفاده کرده و زمان پردازش را افزایش می‌دهد و برای ما که می‌خواهیم به صـــورت آنی پردازش کنیم مفید نیســـت. در عین حال به این خاطر که می‌خواهیم به صورت آنی فرآیند بازسازی را انجام دهیم از همان الگوریتم ORB استفاده می‌کنیم.

روش‌های مطابقت ویژگی که در بخش ۴-۴-۲ توضـــیح داده شـــد در دقتشـــان با یکدیگر تفاوت چندانی ندارند ولی کارایی تطبیقشـــان با هم متفاوت اســـت. به همین منظور ما ارزیابی دقیقی بر روی روش‌های تطبیق FLANN و brute-force انجام نمی‌دهیم. البته روش مورد اســـتفاده در این پایان

نامه روش تطبیق brute-force اســت. برای ارزیابی کلی زمان اجرا، هر مرحله به صــورت جداگانه در جدول زیر لیست شده و میانگین زمان آن‌ها نیز نشان داده شده است.

جدول (۵-۳): زمان اجرا مرحله پردازش برای یک قاب(بر حسب میلی‌ثانیه)

نام مجموعه داده	پیش پردازش	شناسایی ویژگی	تطبیق ویژگی	تخمین موقعیت	زمان به ازای هر قاب
FR1/XYZ	۳۰,۰	۱۶,۰	۱۸,۳	۳۸,۴	۱۰۲,۷
FR1/RPY	۳۲,۲	۱۷,۶	۱۵,۴	۳۶,۴	۱۰۱,۶
FR1/ROOM	۳۱,۱	۱۸,۱	۱۷,۱	۳۹,۲۶	۱۰۵,۵۶
FR1/360	۲۹,۸	۱۳,۳	۱۶,۸	۳۸,۸	۹۸,۷
FR2/PIONEER_360	۳۱,۵	۱۶,۷	۱۹,۰	۴۰,۱	۱۰۷,۳
میانگین	۳۰,۹۲	۱۶,۳۴	۱۷,۳۲	۳۸,۵۹	۱۰۳,۱۷

به طور میانگین برای پردازش هر قاب زمانی در حدود ۱۰۳,۱۷ میلی‌ثانیه نیاز اســـت که متناظر با نرخی در حدود ۱۰ قاب بر ثانیه اســـت. اگرچه این مقدار برای پردازش‌های آنی ممکن اســـت تا حدی مطلوب نباشد ولی برای راه‌کارهایی که مبتنی بر هم‌ترازی ویژگی هستند معمولا نرخ قاب همین مقدار است.

۵-۳ نتایج بازسازی سه‌بعدی محیط داخلی

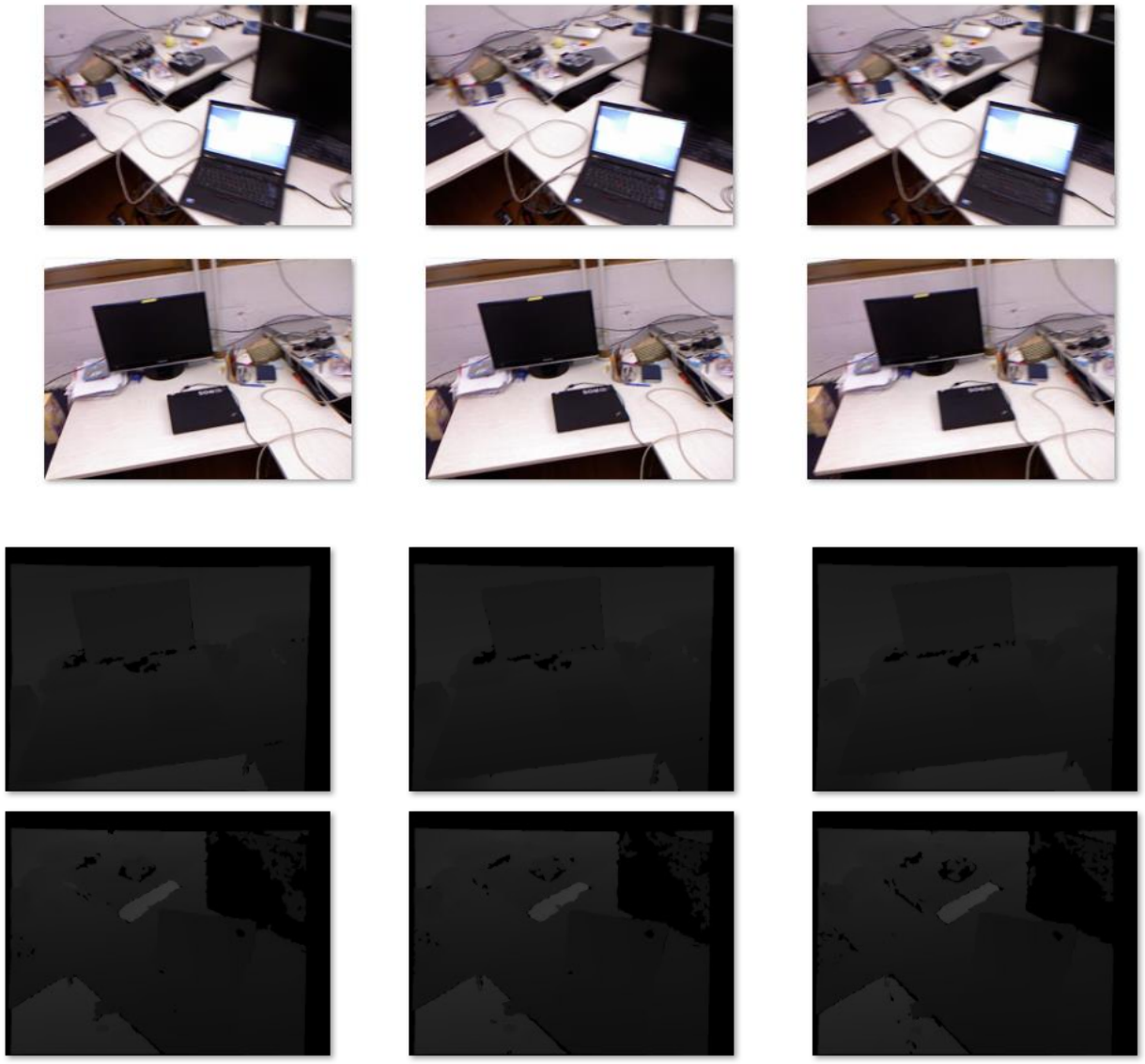
در فصل قبل، ما نحوه استفاده از سیستم را توضیح دادیم و سعی داریم که نتایج را برای مجموعه داده‌ها و داده‌هایی که توسط خودمان بدست آمده، ارزیابی کنیم و صحت و دقت نتایج را بررسی کنیم. همانطور که گفته شد این سیستم برای بازسازی سه‌بعدی محیط داخلی و اشیا قابل استفاده است.

تمام تصاویر توسط سنسور کینکت مایکروسافت با دقت ۴۸۰ × ۶۴۰ که همانطور قبلا گفته شد گرفته شده است. نرخ گرفتن تصویر نیز به علت محدودیت‌های سخت‌افزاری بین ۱۵ تا ۳۰ است.

در بخش بعدی، ما بعضـــی از نماهای ســـه‌بعدی مجموعه داده‌ها را نمایش داده و اندازه بعضـــی از قسمت‌های مدل سه‌بعدی را به وسیله نرم‌افزار MeshLab اندازه‌گیری می‌کنیم.

۵-۳-۱ مجموعه داده FR1/ROOM

این مجموعه داده شامل ۱۳۶۲ تصویر است که توسط سنسور کینکت در دقت ۴۸۰ × ۶۴۰ گرفته شده است. به علت محدودیت‌های پردازشی برای سیستم گفته شده در بخش ۵-۱، ما در حدود ۲۰۰ قاب را برای بازسازی استفاده کرده‌ایم. نقشه سه‌بعدی به دست آمده شامل ۲۰۰حالت دوربین است. در ادامه تعدادی از تصویرهای رنگی و عمق و همچنین نماهای بازسازی شده را می‌توان دید.

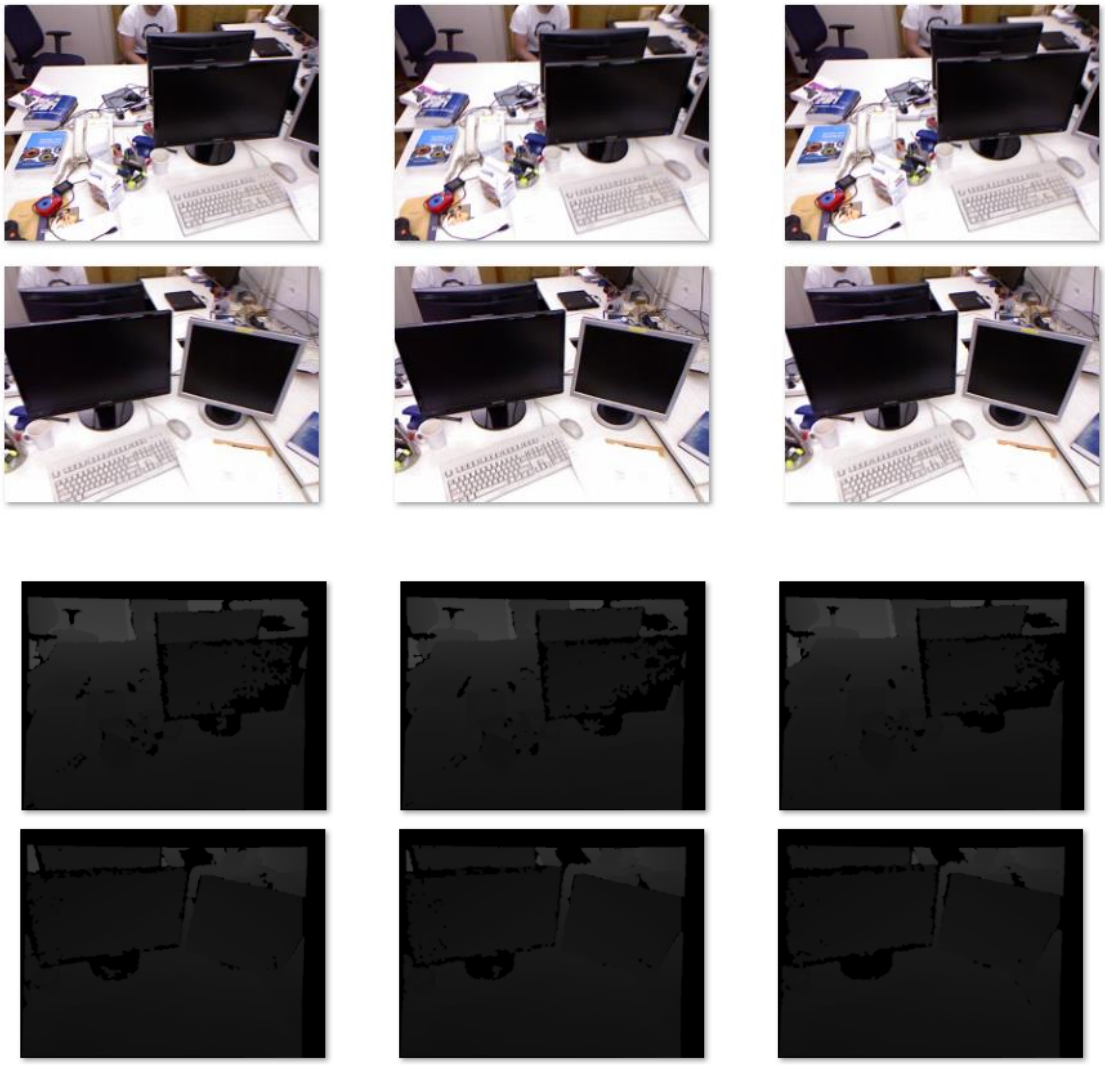


شکل (۵-۳): نماهای متفاوتی از تصاویر رنگی و عمق مجموعه داده FR1/ROOM

شکل (۵-۴): نماهایی از فرآیند بازسازی که با استفاده از ۲۰۰ قاب ساخته شده است

۵-۳-۲ مجموعه داده FR1/XYZ

این مجموعه داده شامل ۷۹۰ قاب است که توسط سنسور کینکت در دقت ۶۴۰ * ۴۸۰ گرفته شده است. نقشه سه‌بعدی به دست آمده شامل ۱۵۰ حالت دوربین است. در ادامه نماهای متفاوتی از این مجموعه داده را می‌بینیم.

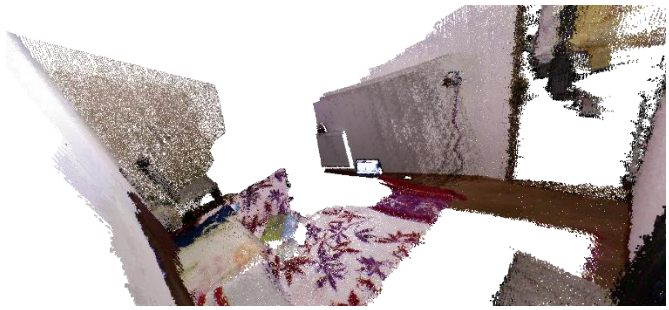


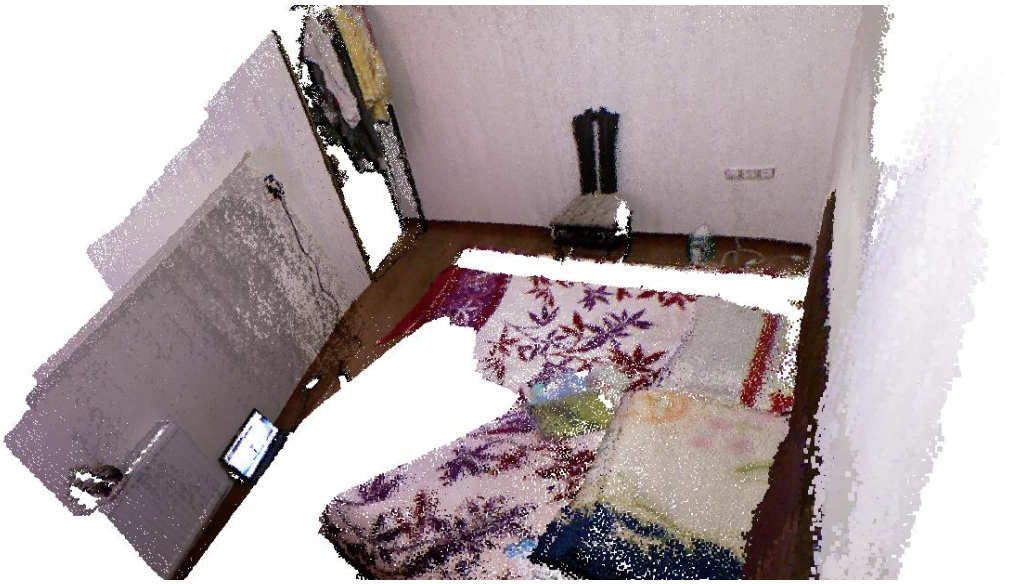
شکل (۵-۵): نماهای متفاوتی از تصاویر رنگی و عمق مجموعه داده FR1/XYZ

شکل (۵-۶): نماهایی از فرآیند بازسازی که با استفاده از ۱۵۰ قاب ساخته شده است

۵-۳-۳ بازسازی سه‌بعدی از محیط داخلی با داده‌های کینکت

در نهایت ما به وسیله کینکت که آن را در محیط حرکت میدهیم قسمتی از یک اتاق را مدل‌سازی کرده و از نماهای متفاوت می‌توانیم جزئیات مختلف اتاق را ببینیم. برای ساخت این مدل سه‌بعدی در حدود ۱۵۰ قاب ۶۴۰ * ۴۸۰ باضـافه اطلاعات عمق از کینکت بدسـت آمده و به برنامه می‌دهیم. این کار به صـورت آنی انجام می‌شـود یعنی به این معنی که کینکت را در محیط به آرامی حرکت داده و ویژگی‌ها را از تصاویر استخراج کرده و در نهایت مدل را به صورت سه‌بعدی بدست می‌آوریم. مدل سه بعدی ساخته شده شامل یک میلیون پیکسل است.





شکل (۵-۷): نماهای متفاوتی از بازسازی سه‌بعدی انجام شده

شکل (۵-۸): اندازه‌گیری عرض تخت خواب توسط نرم‌افزار MeshLab

۵-۴ مقایسه میزان خطا

در پایان برای بررسی میزان خطا، سیستم پیشنهادی در این پایان‌نامه را با روش FOVIS گفته شده در [۱۵] مقایسه می‌کنیم. در روش FOVIS برای باز سازی سه‌بعدی از روش هم‌ترازی سه‌بعدی مبتنی بر ویژگی و ICP استفاده شده است. همچنین این روش برای شناسایی ویژگی‌ها از روش SIFT استفاده می‌کند.

جدول(۵-۴): مقایسه خطای ATE(برحسب متر) روش پیشنهادی با FOVIS

نام مجموعه داده	FOVIS	پیشنهادی
FR1/XYZ	**۰,۰۵۱**	۰,۱۵۷
FR1/RPY	۰,۰۹۷	**۰,۰۵۸**
FR1/ROOM	**۰,۱۸۴**	۰,۴۷۰
FR1/360	۰,۱۴۰	**۰,۱۲۰**
FR2/PIONEER_360	**۰,۰۶۴**	۰,۰۶۵

همانطور که مشاهده می شود سیستم پیشنهادی ما برای مجموعه داده‌هایی که حرکت دوربین آهسته است از روش FOVIS نتایج بهتری را ارائه می‌دهد. البته به علت اینکه سیستم FOVIS از الگوریتم ICP استفاده کرده است در مواردی که حرکت دوربین سریع است نتایج بهتری دارد.

۵-۵ مقایسه زمان اجرای الگوریتم

همانطور که در بخش ۵-۲-۴ گفتیم سیستم پیشنهادی ما قادر است تا ۱۰ فریم بر ثانیه را پردازش کند در حالی که سیستم FOVIS فقط می‌تواند حداکثر تا ۲ فریم بر ثانیه را پردازش کند. در جدول ۵-۵ زمان پردازش انجام شده برای هر مجموعه داده به تفکیک آمده است.

جدول(۵-۵): مقایسه زمان اجرا بر حسب ثانیه

نام مجموعه داده	FOVIS	پیشنهادی
FR1/XYZ	۳۷۱	**۸۵**
FR1/RPY	۵۳۵	**۹۵**
FR1/ROOM	۶۶۰	**۱۲۲**
FR1/360	۳۹۷	**۹۱**
FR2/PIONEER_360	۳۳۳	**۷۰**

با مقایسه نتایج بدست آمده مشاهده می شوند که روش پیشنهادی ما سریع‌تر از روش FOVIS است. با توجه به اینکه روش FOVIS از الگوریتم ICP استفاده می‌کند نتایج نسبتا بهتری را بدست آورده ولی همین موضوع باعث افزایش زمان پردازش شده و فرآیند نقشه‌برداری را کند می‌کند.

فصل ششم: نتیجه گیری و پیشنهادات

۶-۱ مقدمه

در این پایان‌نامه یک سیستم باز سازی سه‌بعدی مبتنی بر RGB-D را توسعه دادیم که به راحتی برای کارهای آنی قابل استفاده است. راه‌کار ما برای بازسازی اشیا بزرگ و همچنین محیط‌های داخلی نسبتا کوچک قابل استفاده است.

برای دستیابی به این هدف، ما ابتدا داده‌های RGB-D را به وسیله سنسور مایکروسافت کینکت که با دست نگاه داشته شده ذخیره می‌کنیم. سپس نقشه عمق بدست آمده توسط سنسور را با فیلتر bilateral پیش پردازش کرده و مقادیری از عمق را که خیلی بزرگ بوده و دارای نویز هستند حذف می‌گردند.

حالت دوربین برای هر قاب به وسیله هم‌ترازی سه‌بعدی مبتنی بر ویژگی تعیین می‌گردد. نقاط ویژگی را که متمایز هستند شناسایی کرده و از تصاویر RGB استخراج می‌کنیم، که این نقاط به ما اجازه تطبیق دو جفت تصویر را می‌دهد. با ترکیب کردن نقاط ویژگی دوبعدی با اطلاعات عمق، تناظر نقاط سه‌بعدی بین تصویرها را بدست می‌آوریم. هم‌ترازی سه‌بعدی مقاوم مبتنی بر RANSAC تطبیق‌های اشتباه را حذف می‌کند و هم‌ترازی سه‌بعدی نسبی بین دو قاب را محاسبه می‌کند.

قسمت نقشه‌برداری، نقشه محیط را که شامل حالت‌های دوربین و ویژگی‌های مشاهده شده است را نگاه‌داری می‌کند. تخمین اولیه برای حالت‌های مطلق دوربین[1] به وسیله ردیابی تصویر به تصویر انجام می‌شود.

به منظور کاهش خطا که به صورت قاب به قاب اضافه می‌شود بهترین ویژگی‌های بدست آمده را استفاده می‌کنیم.

در نهایت با توجه به حالت‌های بهینه بدست آمده نقشه سه‌بعدی کامل را تولید می‌کنیم.

[1] Absolute camera poses

نتایج کار نیز برای مجموعه داده‌های مختلف و همچنین با استفاده از سنسور کینکت خودمان نیز نشان داده شده است. نقشه سه‌بعدی بدست آمده نیز دارای ابعاد واقعی بوده که این مسئله توسط نرم‌افزار MeshLab قابل مشاهده است.

برای بهبود عملکرد سیستم توسعه داده شده، راه‌کارهای متفاوتی وجود دارد که می‌تواند در آینده برای کارایی موثرتر سیستم استفاده شود:

با توجه به اینکه پیدا کردن و شناسایی ویژگی کاری زمان بر است می‌توان از روش‌هایی مانند SiftGPU که از پردازنده گرافیکی استفاده کرده و سرعت آشکارسازی ویژگی را افزایش می‌دهد استفاده کرد.

در حالی که استفاده از RANSAC که به طور مقاوم هم‌ترازی سه‌بعدی را بین نقاط متناظر سه‌بعدی تخمین می‌زند، می‌توان از روش‌های سریعتری مانند PROSAC[11] استفاده کرد.

با توجه به اینکه هم‌ترازی سه‌بعدی دارای خطا است می‌توان از الگوریتم ICP به صورت پردازش موازی، برای افزایش دقت هم‌ترازی بهره برد.

به علت اینکه دقت سنسور ما در مقایسه با دیگر سنسورهای گران قیمت و دقیق پایین است، خطای جمع شونده در هر تصویری که بدست می‌آوریم افزایش پیدا می‌کند. برای اینکه به این مشکل غلبه کنیم می‌توان از روش‌هایی مانند Bundle Adjustment استفاده کرد که خطا را تا حد قابل قبولی کاهش می‌دهد.

مراجع

[1] E. López *et al.*, "A multi-sensorial simultaneous localization and mapping (SLAM) system for low-cost micro aerial vehicles in GPS-denied environments," vol. 17, no. 4, p. 802, 2017.

[2] *Rapidform case study: Speeding the design process for NASA's SIERRA Unmanned Aerial Vehicle (UAV)*. Available: http://www.rapidform.com/success-stories/aerospace-defense/

[3] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *null*, 2003, p. 1403: IEEE.

[4] A. J. Davison, I. D. Reid, N. D. Molton, O. J. I. T. o. P. A. Stasse, and M. Intelligence, "MonoSLAM: Real-time single camera SLAM," no. 6, pp. 1052-1067, 2007.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 2007, pp. 225-234: IEEE.

[6] S. Izadi *et al.*, "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 559-568: ACM.

[7] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 1498-1505: IEEE.

[8] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *In the 12th International Symposium on Experimental Robotics (ISER*, 2010: Citeseer.

[9] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 1691-1696: IEEE.

[10] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held RGB-D camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, 2011, vol. 180, pp. 1-15.

[11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3607-3613: IEEE.

[12] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[13] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[14] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-d vision: from images to geometric models*. Springer Science & Business Media, 2012.

[15] R. Memmesheimer, "Erstellung einer 3-D-Karte aus RGB-D-Daten unter Benutzung visueller Odometrie," Universität Koblenz-Landau, 2015.

[16] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on pattern analysis and machine intelligence*, no. 5, pp. 698-700, 1987.

[17] G. Bradski, "The OpenCV library. Doctor Dobbs Journal, 25(11):120–126, 2000."

[18] A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. " O'Reilly Media, Inc.", 2016.

[19] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 2320-2327: IEEE.

[20] R. A. Newcombe *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, 2011, pp. 127-136: IEEE.

[21] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," ed: Google Patents, 2012.

[22] *Asus Xtion Pro Live product webpage*. Available: http://www.asus.com/Multimedia/Xtion_PRO_LIVE/

[23] J.-Y. Bouguet, "Camera calibration toolbox for MATLAB," 2004.

[24] M. Kinect, "https://www.microsoft.com/en-us/download/details.aspx?id=40278," 2009.

[25] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, 1992, vol. 1611, pp. 586-607: International Society for Optics and Photonics.

[26] T. Tykkälä, C. Audras, and A. I. Comport, "Direct iterative closest point for real-time visual odometry," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011, pp. 2050-2056: IEEE.

[27] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011, pp. 719-722: IEEE.

[28] C. Harris and M. Stephens, "A combined corner and edge detector in Alvey vision conference. 1988," *Manchester, UK*.

[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91-110, 2004.

[30] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)(2007)," *URL http://cs. unc. edu/~ ccwu/siftgpu*, 2011.

[31] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, 2006, pp. 404-417: Springer.

[32] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE international conference on*, 2011, pp. 2564-2571: IEEE.

[33] D. L. Baggio, *Mastering OpenCV with practical computer vision projects*. Packt Publishing Ltd, 2012.

[34] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[35] M. A. Fischler and R. C. J. C. o. t. A. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," vol. 24, no. 6, pp. 381-395, 1981.

[36] V. Högman, "Building a 3D Map from RGB-D Sensors," ed, 2012.

[37] O. Chum and J. Matas, "Matching with PROSAC-progressive sample consensus," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, vol. 1, pp. 220-226: IEEE.

[38] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 573-580: IEEE.

[39] B. K. J. J. A. Horn, "Closed-form solution of absolute orientation using unit quaternions," vol. 4, no. 4, pp. 629-642, 1987.

Abstract

In this thesis, we have developed a 3D reconstruction system based on RGB-D data. The proposed approach is suitable for both 3D object modeling and indoor reconstruction.

First, we record the RGB-D data by a Kinect sensor that is held by hand. The depth of the sensor is refined in the pre-processing stage and its noise is partially eliminated. The external parameters of the camera are obtained in motion between two frames (including the rotational matrix and displacement vector), and then track the motion of the camera using a using a robust 3D alignment algorithm based on RANSAC. Then, we check the camera movement rate to increase the accuracy of mapping. The basic map of the environment, which includes camera poses between the two frames, features and observations of the scene, is made using frame tracking to the frame and added to the overall map.

In order to reduce memory consumption and improve processing time, instead of storing information from all 3D cloud pixels, we use voxels and reduce the cloud-point volume.

Finally, the kinect sensor for each pixel gives the exact depth, the indoor 3D map is an in-depth metric scale, which can be useful for in-depth visual inspections, reverse engineering and measurement processes.

Keywords: 3D Reconstruction, 3D Modeling, Kinect

Shahrood University of Technology

Faculty of Mechanical and Mechatronics Engineering

M.Sc. Thesis in Mechatronics Engineering

Indoor 3D modeling by Kinect sensor

By: Mohamad Javad Tahmasebi

Supervisor:

Dr. Hosein khosravi

September 2018