دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف

محاسبه‌ی کارای فضای قابل دید و ساده‌سازی آن در محیط‌های مختلف

رساله ارائه شده به عنوان بخشی از ملزومات برای دریافت درجه
دکتری کامپیوتر-نرم‌افزار

نگارش
علیرضا زارعی

استاد راهنما
دکتر محمد قدسی

آبان ۱۳۸۷

تقدیم به

مادر مهربان و همسر عزیزم

قدردانی

خداوند را شاکرم که این فرصت را برای من فراهم نمود تا گامی هرچند ناچیز در راستای پیشبرد اهداف جامعه علمی بردارم و سپاسگذارم که در طول این دوره، معجزه‌وار مرا یاری نمود و به من لیاقت داد تا بتوانم به این مهم دست یابم.

از استاد گرامی جناب آقای دکتر محمد قدسی کمال تشکر و قدردانی را دارم. بی‌شک، راهنمایی‌ها و همکاری‌های بی‌دریغ ایشان مهم‌ترین عامل در دستیابی به نتایج حاصل بوده است و با وجود تمام کاستی‌ها و نواقصی که در برنامه تحقیقی من وجود داشت، با مدیریت و سرپرستی مناسب خود دستیابی به اهداف این دوره تحقیقی را میسر نمودند. برای ایشان سلامتی و موفقیت هرچه بیشتر را آرزومندم.

از دکتر مارک دی‌برگ از دانشگاه صنعتی آیندهوون هلند تشکر می‌کنم که این امکان را فراهم آوردند تا در یک دوره سه ماهه در کنار گروه تحقیقاتی ایشان کار کنم و نتایج موثری بدست بیاورم. همچنین از کلیه اعضای گروه ایشان به ویژه دکتر محمدعلی آبام، محمد فرشی و پیتر هاخنبرگر تشکر می‌کنم که صمیمانه مرا به عنوان یک همکار در جمع خود پذیرفتند.

جا دارد از دکتر رامتین خسروی به خاطر راهنمایی‌ها و مشاوره‌های موثرشان تشکر کنم که تجربیات دوره خود را در اختیار من قرار دادند و این تجارب به عنوان علائم و چراغ‌هایی همواره در مسیر تحقیقاتی کمک و راهنمای من بودند.

از کلیه کسانی که به عنوان نویسنده همکار در تهیه و چاپ نتایج تحقیقی همکاری داشته‌اند از جمله دکتر محمد قدسی، دکتر مارک دی‌برگ، دکتر محمدعلی آبام، دکتر پیتر هاخنبرگر، علی خسروی و مجتبی نوری‌بایگی تشکر می‌کنم.

همچنین، از مسوولین و اساتید گروه کامپیوتر و دانشجویان آزمایشگاه دکتر قدسی در دانشکده مهندسی کامیوتر دانشگاه صنعتی شریف، دانشجویان و محققان گروه کامیوتر پژوهشکده دانش‌های بنیادی و فیزیک نظری و گروه الگوریتم دانشگاه آیندهوون که در انجام این تحقیق موثر بوده‌اند تشکر می‌کنم.

از مدیران و همکاران خود در گروه شرکت‌های همکاران سیستم به خاطر همکاری و پشتیبانی‌هایشان سپاس‌گذاری می‌کنم.

درپایان، از کمک، تشویق و صبر و شکیبایی همسرم بسیار سپاس‌گذارم که در برهه مهمی از این دوره تحقیقی در کنار من بودند و برای به ثمر رسیدن آن از هیچ کمکی دریغ نکردند.

محاسبه‌ی کارای فضای قابل دید

و ساده‌سازی آن در محیط‌های مختلف

چکیده

در این رساله، گونه‌های مختلفی از مسائل قابلیت دید مطالعه شده‌اند. این مسائل شامل محاسبه چندضلعی قابل دید ناظر نقطه‌ای در یک دامنه چندضلعی گونه، نگهداری چندضلعی قابل دید یک ناظر نقطه‌ایِ متحرک در یک دامنه چندضلعی گونه، نگهداری چندضلعی قابل دید از یک ناظر پاره‌خط‌ای متحرک، ارتباطات دیداری در فضای سه‌بعدی و ساده‌سازی مرز ناحیه قابل دید است. ما این مسائل را برای دو حالت جویباری و غیرجویباری مطالعه کرده‌ایم.

این مسائل دارای کاربرد در حوزه‌های مختلفی از جمله گرافیک کامپیوتری، دید ماشین، برنامه‌ریزی حرکت، سیستم‌های اطلاعاتی جغرافیایی و هندسه محاسباتی است.

نتایج بدست آمده هم شامل الگوریتم‌های نظری و هم شامل روش‌های کاربردی برای حل این مسائل است. این نتایج را می‌توان به دو دسته‌ی محاسبه و نگهداری دید، و ساده‌سازی دید تقسیم کرد. نتایج بدست آمده در دسته اول شامل الگوریتم‌های کارایی برای محاسبه چندضلعی قابل دید از ناظر نقطه‌ای در دامنه‌های چندضلعی گونه، نگهداری چندضلعی قابل دید از ناظر نقطه‌ای یا پاره‌خط‌ای متحرک در دامنه‌های چندضلعی گونه و تعیین روابط دیداری بین اشیاء در فضای سه‌بعدی است. نتایج مربوط به رده دیگر شامل تعیین معیارهای ساده‌سازی وابسته به ناظر برای ناظرهای نقطه‌ای و پاره‌خط‌ای در صفحه و ارائه الگوریتم‌های ساده‌سازی برای ساده‌سازی مرز ناحیه قابل دید از این ناظرها در دو حالت جویباری و غیرجویباری است. به عنوان یک نتیجه جانبی، یک الگوریتم ساده‌سازی عمومی نیز بدست آمده است که برای ساده‌سازی در حالت جویباری برای کلیه مسائل (علاوه‌بر مسائل قابلیت دید) قابل استفاده است.

واژه‌های کلیدی: هندسه محاسباتی، مسائل قابلیت دید، ساده‌سازی مسیر، الگوریتم‌های جویباری

فهرست مطالب

۱ مقدمه

۲ مفاهیم، داده‌ساختارها و الگوریتم‌های پایه

۷ مجتمع قابلیت دید سه‌بعدی

۸ ساده‌سازی چندضلعی قابل دید

۹ ساده‌سازی در مدل جویباری

۱۰ نتیجه‌گیری

فهرست شکل‌ها

فصل ۱

مقدمه

تعیین دید یکی از مسائل بنیادی در کاربردهای گوناگون روزمره است. این مساله به طور غیررسمی به صورت تعیین مجموعه نقاط قابل دید از یک ناظر تعریف می‌شود. شکل هم‌ارز دیگر این مساله، تعیین فضایی است که توسط یک منبع نورانی روشن می‌شود. در هریک از حوزه‌های کاربردی مسائل قابلیت دید، علیرغم یکسان بودن هدف، روش‌ها و اولویت‌های مختلفی وجود دارد که در ادامه به اختصار معرفی می‌شوند.

علاوه‌بر کاربرد، پیچیدگی و متنوع بودن مسائل قابلیت دید دلیل دیگری برای توجه محققان زیادی به این مساله است. در این بخش، ابتدا برخی از شکل‌های مختلف مسائل قابلیت دید در حوزه‌های مختلف و طبقه‌بندی روش‌های حل این مسائل بیان می‌شود. سپس انگیزه انتخاب این دسته از مسائل در تدوین این پایان‌نامه و نتایج بدست آمده ذکر می‌شود.

۱-۱ مسائل قابلیت دید

مسائل قابلیت دید در حوزه‌های کاربردی مختلفی مطرح هستند. در هریک از این حوزه‌ها دسته‌هایی از مسائل مطرح می‌شود که بطور مستقیم به مسائل قابلیت دید مربوط می‌شوند. این حوزه‌ها شامل گرافیک کامپیوتری، دید ماشین، روباتیک و هندسه محاسباتی است که در زیر به آنها می‌پردازیم.

۱-۱-۱ گرافیک کامپیوتری

تحقیقات گسترده‌ای در حوزه گرافیک کامپیوتری روی مسائل قابلیت دید انجام شده و نتایج زیادی بدست آمده است [۴۷، ۱۱۸، ۱۳۲]. در این‌جا برخی از مسائل مطرح در گرافیک که در آنها مسائل قابلیت دید اهمیت دارد بیان می‌شود.

حذف سطوح مخفی

محاسبه دید اصلی‌ترین مساله در تحقیقات ابتدایی گرافیکی بوده است. در این حوزه، دید به معنای تعیین بخش‌ها، سطوح یا خطوطی از صحنه است که از یک ناظر قابل مشاهده است. طبقه‌بندی مناسبی از این روش‌ها در [۵۹، ۱۲۵] ارائه شده است. همچنین نتایج نظری مربوط به این حوزه در [۲۰، ۳۶] جمع‌آوری شده است. حالت‌های خاصی از این مساله از جمله دید بلادرنگ در شبیه‌سازی پرواز [۹۹] و نیز نقاشی یا فیلم‌های کارتونی [۹۳، ۱۳۵] بررسی شده است.

محاسبه سایه

محاسبه کارا و واقعی سایه هنوز یکی از مسائل چالش‌برانگیز در گرافیک کامپیوتری است. این مساله در کاربردهای واقعی و سه‌بعدی اهمیت ویژه‌ای دارد. کتاب باگزاندال[۱] جنبه‌های جالبی از مفهوم سایه را در نقاشی، فیزیک و کامپیوتر مطرح کرده‌است [۱۸]. مفهوم سایه کامل[۲] که معادل با فضایی از صحنه است که از هیچ بخشی از منبع نور دیده نمی‌شود معادل با محاسبه دید یا حذف سطوح مخفی است. ولی نیم‌سایه[۳] مساله پیچیده‌تری است که محاسبه شدت نور در هر نقطه آن مطرح است. روش‌هایی که برای حل این مساله ارائه شده است در [۱۳۶] جمع‌آوری شده است. مساله جالب دیگر، وارون سایه است که به معنای تشخیص منبع نور با داشتن سایه است که بسیار کم به آن پرداخته شده است.

تعیین موانع

جاهایی که پیچیدگی صحنه زیاد باشد، روشهای رسم تعاملی حتی بر روی سخت‌افزارهای قوی هم جوابگو نیست. به عنوان مثال، حرکت از بین سطوح سه‌بعدی در شبیه‌سازی پرواز یا رانندگی و رسم نقشه‌های پیچیده CAD/CAM با روش‌های ابتدایی گرافیکی قابل حل نیست. در مساله تعیین موانع، سعی می‌شود که اشیائی که در دید قرار ندارند مشخص شده و به دستگاه نمایشگر داده شوند که از رسم و پردازش آنها اجتناب شود. این روش‌ها به دسته‌های برخط[۴] و غیر برخط[۵] تقسیم می‌شوند. در الگوریتم‌های برخط به ازای هر بازنمایش تصویر، اشیای غیر قابل دید مشخص می‌شوند در حالی‌که در روش‌های غیربرخط، اشیائی که در هر موقعیت مکانی ناظر ممکن است دیده شوند تعیین و در زمان نمایش تنها این اشیاء رسم می‌شوند [۳۲، ۷۰].

---

[۱] Baxandall

[۲] Hard Shadow

[۳] Soft Shadow

[۴] Online

[۵] Offline

نورپردازی سراسری

در نورپردازی سراسری، شبیه‌سازی نور براساس قوانین فیزیکی (انعکاس و جذب) و تعامل نوری بین اشیاء صورت می‌گیرد. در این مساله، نور توسط اشیای آینه‌ای منعکس می‌شود؛ اشیای شفاف باعث عبور نور می‌شوند؛ اشیای کدر باعث بوجود آمدن سایه می‌شوند و برخی اشیاء باعث انعکاس نور در همه جهت‌ها می‌شوند. این مساله در مورد انتشار صدا و سایر امواج نیز صدق می‌کند. نتایج بدست آمده برای این مسائل در مراجع [۳۳، ۳۴، ۴۸، ۵۶، ۸۱، ۸۷، ۱۲۲] آمده است.

ردیابی اشعه

در این مساله، مسیری که یک اشعه طی می‌کند دنبال می‌شود. این مساله حالت خاصی از نور پردازی است و با توجه به بازگشتی و نامتناهی بودن آن بصورت احتمالی حل می‌شود [۱۰، ۱۲، ۵۵، ۶۵، ۱۱۴، ۱۳۰، ۱۳۴]. مهمترین مساله در این کاربرد تعیین نخستین شیئی است که در امتداد یک شعاع نوری قرار دارد.

Radiosity

این مساله ابتدا در مورد حرارت تعریف [۲۵] و سپس به نور تعمیم داده شد. در این مساله، با این فرض که همه اشیاء نور دریافت کرده را در همه جهات منعکس می‌کنند، می‌خواهیم درخشندگی یا تاری هر نقطه از اشیاء را مشخص کنیم. این مساله معادل با کاربردهای واقعی است که در یک اتاق، گوشه‌ها تاریک‌تر به نظر می‌رسند. در این مساله، میزان روشنایی هر نقطه براساس جنس آن و زاویه‌ای که با منبع نور و نقطه دید می‌سازد، تعیین می‌شود [۱۶، ۳۳، ۱۲۲].

۱-۱-۲ دید ماشین

نمونه‌های متعددی از این مسائل مطرح شده است [۴۵، ۶۰، ۱۷] که در زیر برخی از آنها را بیان می‌کنیم.

تشخیص شی

در این مساله، پایگاه‌داده‌ای از اشیای شناخته شده موجود است و می‌خواهیم با مشاهده یک عکس، شی مربوط را تشخیص دهیم. برای این‌کار لازم است که داده‌ساختار مناسبی از تصویر هر شی در جهت‌های مختلف موجود باشد و با تطبیق تصویر داده‌شده با این تصاویر، شی را شناسایی کنیم [۱۱۰].

بازسازی شی

بازسازی شی به معنی ساختن مدل سه‌بعدی یک شی از روی مجموعه‌ای از تصاویر آن است که از زاویه‌های دید مختلف تهیه شده‌اند. نکته‌ای که در این مساله وجود دارد این است که تا چه اندازه یک شی خودش را پوشانده است. در صورتی که تعداد تصاویر کافی موجود باشد این مساله بصورتی کارا قابل حل است [۱۱۱، ۱۲۳].

برنامه‌ریزی نقطه دید

بدست آوردن اطلاعات درست و با کیفیت از یک محیط به تعیین مناسب نقاطی که سنسورها در آنجا قرار می‌گیرند بستگی دارد. این مساله برنامه‌ریزی نقطه دید یا مکان‌یابی سنسور گفته می‌شود. این مساله در کاربردهای مختلفی مانند کنترل کیفیت و نگهبانی تعریف و برای آن راه‌حل ارائه شده است [۷۸، ۱۱۷، ۱۲۷].

کشف محیط

در برخی کاربردها، می‌خواهیم یک شی ناشناخته را بطور کامل شناسایی کنیم. اینکه نقطه دید در چه مکان‌هایی قرار بگیرد تا درک بهتری از این شی داشته باشد، یک مساله کاربردی و جالب است [۴، ۱۱۷، ۱۲۷].

۱-۱-۳ روباتیک

مسائل قابلیت دید بسیار زیادی در حوزه روباتیک مطرح هستند که مراجع [۶۷، ۶۸، ۸۸] مهمترین آنها را تشریح کرده‌اند.

برنامه‌ریزی حرکت

یک ربات واقع در یک نقطه از صفحه را در نظر بگیرید که می‌خواهد از میان موانع عبور کند و به نقطه دیگری برسد. از آنجا که یک ربات در امتداد یک خط راست تنها می‌تواند تا نقاط قابل دید خود حرکت کند، برنامه‌ریزی بهینه و درست این حرکت بستگی زیادی به محاسبه و استفاده از اطلاعات قابلیت دید محیط دارد. گونه‌های مختلفی از این مساله مطرح و برای آنها راه حل‌هایی ارائه شده است [۸۸].

تعقیب و گریز

یک نمونه از مسائل برنامه‌ریزی حرکت، تعقیب و یافتن یک هدف در محیط است. تعقیب کننده بخشی از صحنه را که می‌بیند پاک می‌کند و این بخش تا زمانی که هدف وارد آن نشود پاک باقی می‌ماند. نحوه حرکت تعقیب کننده بطوری که بتواند کل محیط را پاک کند و هدف را پیدا کند مساله‌ای است که ارتباط مستقیمی با مسائل قابلیت دید دارد.

خود مکان‌یابی

در این مساله، یک روبات واقع در یک صحنه، با داشتن نقشه صحنه می‌خواهد موقعیت خود را در صحنه تشخیص دهد. این مساله شباهت زیادی به مساله تشخیص شی در حوزه دید ماشین دارد. یک روش برای حل این مساله در [۳۹] ارائه شده است.

۱-۱-۴ هندسه محاسباتی

مفاهیم و مطالب پایه‌ای هندسه محاسباتی در کتاب‌های [۲۲، ۲۳، ۴۲، ۱۰۵] بیان شده‌اند. بسیاری از مسائل این حوزه مربوط به حل نظری مسائل روباتیک است. با اینکه ارائه الگوریتم‌هایی که بصورت نظری بهینه هستند در اولویت این حوزه قرار دارد ولی فعالیت‌های مربوط به پیاده‌سازی این الگوریتم‌ها نشان از توجه محققین این حوزه به کاربرد و قابل پیاده‌سازی بودن الگوریتم‌های ارائه شده دارد [۳۰، ۹۲، ۱۲۶].

تقریبا همه مسائل حوزه‌های یاد شده قبلی در این حوزه نیز از لحاظ نظری بررسی شده‌اند.

حذف سطوح مخفی

این مساله گرافیکی بصورتی گسترده در هندسه محاسباتی مطالعه و بررسی شده است. برای حالت سه‌بعدی آن الگوریتمی بهینه با پیچیدگی زمانی $O(n^2)$ ارائه شده است [۹۴]. گونه‌های مختلفی از این مساله تعریف و حل شده‌اند که در این رساله نیز به حل مسائلی در این حوزه می‌پردازیم.

پرتوافکنی

اینکه یک شعاع نوری پس از انتشار از یک منبع نوری، ابتدا به کدام شی برخورد می‌کند مساله‌ای است که توجه زیادی در حوزه هندسه محاسباتی به خود جلب کرده است. حالت کلی این مساله وضعیت خطوط در فضا است که برای کاربردهای مختلف بررسی شده است [۱۰۷].

موزه هنر

این سوال که برای نگهبانی یک موزه چند نگهبان لازم است در سال ۱۹۷۳ مطرح شد و پس از آن گونه‌های مختلفی از این مساله تعریف و حل شده‌اند [۱۰۴، ۱۲۱، ۱۲۹]. واضح است که برای حل این مساله باید بتوانیم فضای قابل دید از هر نگهبان را به عنوان یک زیرمساله تعیین کنیم.

داده ساختارهای قابلیت دید

داده ساختارهای مختلفی شامل گراف قابلیت دید[۶]، گراف جنبه[۷] و مجتمع قابلیت دید[۸] برای حل مسائلی از جمله کوتاه‌ترین مسیر و محاسبه دید ارائه شده است. در این داده‌ساختارها روابط دیداری بین اشیاء نگهداری می‌شود.

۱-۱-۵ جمع‌بندی مسائل

براساس طبقه‌بندی صورت گرفته در [۵۹]، مسائل قابلیت دید را می‌توان براساس افزایش بعد طبقه‌بندی کرد: نخستین مساله پرتوافکنی است که حالت یک‌بعدی دید را نشان می‌دهد. پس از آن، محاسبه دید و سایه کامل گونه‌های دوبعدی این مساله هستند. تعیین موانع نسبت به یک نقطه نیز در همین دسته قرار می‌گیرند. سپس مسائل قابلیت دید سراسری قرار دارند که دید را نسبت به منابع نوری غیرنقطه‌ای و در فضای سه‌بعدی مشخص می‌کنند. همچنین، تعیین بخشی از فضا که شی خاصی را می‌بیند در این دسته قرار دارد. تعیین تناظر دیداری بین اشیاء که در نورپردازی سراسری مطرح است از جمله مسائل قابلیت دید چهاربعدی است. نهایتا، مسائل مربوط به شمارش تعداد نقاط دید برای شناسایی یک شی یا بهینه‌سازی نقاط دید قرار دارند.

۱-۲ طبقه‌بندی روش‌ها

الگوریتم‌های ارائه شده برای حل مسائل قابلیت دید به دو دسته فضای شی[۹] و فضای تصویر[۱۰] تقسیم می‌شوند [۱۲۵]. این تقسیم‌بندی مبتنی بر دقت انجام محاسبات است.

---

[۶] Visibility Graph

[۷] Aspect Graph

[۸] Visibility Complex

[۹] Object-Space

[۱۰] Image-Space

فضای تصویر

تعداد زیادی از الگوریتم‌ها، محاسبه دید را براساس تصویر اشیاء بر روی یک صفحه دوبعدی انجام می‌دهند. این صفحه لزوما صفحه نمایش که تصویر در آن قرار می‌گیرد نیست و ممکن است یک صفحه مجازی و میانی باشد. درصورتی که صحنه دوبعدی باشد این صفحه تصویر به یک بعد تقلیل می‌یابد که محدود به زاویه نقطه دید است. این الگوریتم‌ها معمولا با حالت گسسته و دودویی از صفحه که دقت آنها محدود است کار می‌کنند و دقت تصویر بدست آمده محدود به دقت این صفحه است.

فضای شی

در مقابل، برای دسته دیگری از الگوریتم‌ها، فضای دید همان فضای صحنه است که براساس دقت اشیای موجود در صحنه، فضای قابل دید تعیین می‌شود.

برخی از الگوریتم‌ها مبتنی بر ترکیبی از این دو حالت هستند. الگوریتم‌هایی که در حوزه هندسه محاسباتی مطرح می‌شوند مبتنی بر فضای شی هستند و در مقابل اغلب الگوریتم‌های کاربردی و مطرح در گرافیک کامپیوتری براساس فضای تصویر عمل می‌کنند. با توجه به اینکه این رساله در حوزه هندسه محاسباتی به مسائل قابلیت دید می‌پردازد، در فضای شی مسائل را بررسی و مطالعه می‌کند.

۱-۳ انگیزه

دو انگیزه مختلف و مکمل در تدوین این رساله دخیل بوده‌اند: نظری و کاربردی. محاسبه دید در روش‌های زیادی مطرح می‌شود و زمان این محاسبه گلوگاه سرعت اجرا در این کاربردها است. در عین حال، خود محاسبه دید شامل مسائل هندسی جذابی است که هنوز به اندازه کافی مورد مطالعه قرار نگرفته‌اند.

از دید نظری هنوز فاصله‌هایی بین زمان بهینه محاسبه دید و زمان اجرای بهترین الگوریتم در برخی مسائل وجود دارد. برای از بین بردن این فاصله باید الگوریتم‌های بهتری نسبت به روش‌های موجود ارائه شود.

از نظر کاربردی نیز علیرغم راه‌حل‌های فراوان موجود، ارتباط بسیار کمی بین روش‌های کارای نظری و کاربردهای واقعی وجود دارد. به عنوان بخشی از این رساله، سعی شده‌است که الگوریتم‌هایی با معیار کارایی نظری و معیار سادگی پیاده‌سازی در کاربرد ارائه شود.

۱-۴ نتایج بدست آمده

نتایج بدست آمده در این رساله در قالب چندین مقاله گردآوری شده‌اند که فهرست آنها در ضمیمه A آمده است. در این بخش این نتایج را به اختصار شرح می‌دهیم.

محاسبه چندضلعی قابل دید ناظر نقطه‌ای

اولین مساله‌ای که مورد مطالعه قرار گرفت تعیین چندضلعی قابل دید، $V(q)$، از یک ناظر نقطه‌ای در دامنه‌های چندضلعی‌گونه n راسی با h مانع بود. هدف از این مساله تهیه داده‌ساختارهای مناسب در مرحله پیش‌پردازش برای افزایش کارایی در محاسبه دید نقاط پرس‌وجو است. برای این مساله، با زمان پیش‌پردازش $O(n^3 \log n)$ و حافظه $O(n^3)$ توانستیم $V(q)$ را در زمان $O((1+h')\log n + |V(q)|)$ برای نقطه پرس‌وجوی دلخواه q محاسبه کنیم که $h' \leq \max(h, |V(q)|)$. نتایج حاصل از این کار در [۱۴۴، ۱۳۹، ۱۴۶] منتشر شده‌اند. این نتایج نسبت به نتایج موجود از نظر زمان پرس‌وجو برتری دارند.

نگهداری ساختار ترکیبیاتی دید برای ناظر نقطه‌ای متحرک

پس از تعیین فضای قابل دید برای یک نقطه پرس‌وجو، قدم بعدی استفاده از این روش برای محاسبه و نگهداری فضای قابل دید از یک ناظر نقطه‌ای متحرک در دامنه‌های چندضلعی‌گونه بود که حاصل آن یک روش بهینه از نظر تعداد وقایع پردازش شده و نیز سرعت بهینه برای پردازش وقایع است [۱۴۷]. علیرغم هزینه پیش‌پردازش بالاتر، این روش دارای هزینه پرس‌وجوی بهتری نسبت به سایر روش‌های موجود برای حل این مساله است. علاوه‌برآن، در این روش فقط رخدادهای ضروری پردازش می‌شوند که از این نظر نیز نسبت به الگوریتم‌های موجود برتری دارد.

نگهداری چندضلعی قابل دید دقیق برای ناظر نقطه‌ای متحرک

مساله دیگری که به آن توجه شد نگاهی کاربردی به مساله قابلیت دید است که برخلاف روش‌های نظری، روشی را برای نگهداری فضای قابل دید از یک ناظر نقطه‌ای ارائه می‌دهد که با توجه به نحوه نمایش دید در کاربردهای واقعی از جمله پویانمایی یا بازی‌های رایانه‌ای، امکان نگهداری و بهنگام‌سازی کارای فضای قابل دید را فراهم می‌آورد. برخلاف روش‌های صرفا نظری، ملاحظاتی در کاربردها اهمیت دارند که به صورت نظری کمتر به آنها پرداخته می‌شود. یکی از این موارد بهنگام‌سازی پیوسته فضای قابل دید برای یک ناظر متحرک است، درحالی‌که، در الگوریتم‌های نظری تنها جاهایی که ساختار ترکیبیاتی دید تغییر می‌کند مورد توجه قرار می‌گیرد. برای بکارگیری

روش‌های نظری در کاربردهای واقعی باید حالت‌هایی که دید تغییر می‌کند ولی این تغییر منجر به تغییر در ساختار ترکیبیاتی منظره نمی‌شود نیز مورد توجه قرار بگیرد. نکته قابل توجه این است که در نظر گرفتن این حالات باعث کاهش هزینه در روش‌های نظری نیز می‌شود. آنچه که در این راستا انجام شده است استفاده از نمایش خاصی از گراف قابلیت دید است که به کمک آن، بدون نیاز به نگهداری صفی از وقایع آینده (که در همه روشهای نظری بکار می‌روند)، می‌توان منظره قابل دید را بصورت پیوسته بهنگام کرد. نتایج این روش مبتنی بر تهیه و نگهداری گراف قابلیت دید بصورتی ویژه است که اطلاعات دقیق‌تری از وضعیت دید صحنه ارائه می‌دهد. این نمایش خاص از گراف قابلیت دید به همراه اطلاعات اضافه‌ای که در آن تعبیه شده است، بهنگام‌سازی پیوسته منظره دید را میسر می‌کند. در این روش از روش‌های بهینه مربوط به محاسبه گراف قابلیت دید و تعیین فضای قابل دید در چندضلعی‌های با مانع استفاده شده است. در الگوریتم ارائه شده، پس از محاسبه مقدار اولیه چندضلعی قابل دید، در فواصل زمانی متوالی و به صورت منظم دید ناظر متحرک بهنگام می‌شود. در این الگوریتم، بهنگام‌سازی دید در زمان بهینه خطی نسبت به تعداد تغییرات دید انجام می‌شود. علاوه‌برآن، در این الگوریتم برخلاف روش‌های قبلی، هیچ نوع واقعه‌ی غیرضروری پردازش نمی‌شود و تنها وقایعی که باعث تغییر دید می‌شوند پردازش می‌شوند [۱۳۸، ۱۴۰]. زمان و حافظه پیش‌پردازش این الگوریتم برابر است با $O(n^2)$ و بدون نیاز به صف رخدادها، در صورتی که فاصله زمانی درخواست محاسبه $V(q)$ به اندازه کافی کوچک باشد، مقدار دقیق $V(q)$ در زمان خطی، نسبت به تعداد تغییرات $V(q)$ از زمان درخواست قبلی تاکنون، محاسبه می‌شود.

نگهداری چندضلعی قابل دید قوی و ضعیف برای ناظر پاره‌خطی متحرک

مساله دیگری که به آن پرداخته شد نگهداری فضای قابل دید برای یک ناظر متحرک غیرنقطه‌ای است. برای این نوع ناظرها دو گونه تعریف از فضای قابل دید یعنی فضای قابل دید ضعیف و قوی وجود دارد. این مساله برای این دو گونه از فضای قابل دید و برای ناظری که بصورت یک پاره‌خط است و در درون یک چندضلعی ساده حرکت می‌کند با استفاده از داده‌ساختارهای مربوط به نگهداری درخت کوتاه‌ترین مسیر بصورت بهینه حل شد [۸۴]. زمان و حافظه مرحله پیش‌پردازش این الگوریتم به ترتیب برابر است با $O(n \log n)$ و $O(n)$ و هر رخداد تغییر دید برای حالتی که ناظر در امتداد خط راست حرکت می‌کند در زمان $O(\log n)$ پردازش می‌شود و برای حالتی که جهت حرکت ناظر ثابت نباشد، هر رخداد در زمان $O(\log^2 n)$ پردازش می‌شود.

ساده‌سازی چندضلعی قابل دید ناظر نقطه‌ای و پاره‌خطی

یکی دیگر از ملاحظات کاربردی مربوط به فضای قابل دید یک ناظر، ساده‌سازی یا تقریب زدن چندضلعی قابل دید با منحنی دیگری است که پیچیدگی کمتری نسبت به چندضلعی اولیه داشته باشد و در ضمن، تقریب مناسبی از آن نیز باشد. برای این منظور باید معیاری تعریف شود که به

کمک آن بتوانیم میزان شباهت چندضلعی اولیه را با چندضلعی تقریبی بدست آوریم. در این معیار باید فاصله نقاط روی چندضلعی اولیه و تقریب بدست آمده تا نقطه دید در نظر گرفته شود بطوری‌که نقاطی که به ناظر نزدیک‌ترند نسبت به نقاطی که فاصله بیشتری با ناظر دارند از اهمیت بیشتری برخوردار باشند. به همین منظور، دو معیار خطای خاص برای این نوع تقریب ارائه شده است [۱۴۱، ۱۴۳]. این معیارهای خطا، بدون افزایش زمان اجرا یا حافظه مصرفی، قابل استفاده در روش‌های موجود برای مساله ساده‌سازی مسیر[۱۱] هستند. بااین‌حال، این الگوریتم‌ها دارای زمان و حافظه مصرفی درجه دو یا بیشتر هستند. برای رفع این نقیصه یک الگوریتم با زمان اجرا و حافظه مصرفی $O(n^{۴/۳+\delta})$ ارائه کرده‌ایم که می‌تواند برای ساده‌سازی مسیر دید براساس یکی از معیارهای خطای پیشنهاد شده استفاده شود [۱۴۱].

معیار خطای پیشنهاد شده، برای ساده‌سازی چندضلعی قابل دید ضعیف و قوی یک پاره‌خط نیز توسعه داده شده است و با همان هزینه زمانی و حافظه‌ای برای ساده‌سازی چندضلعی قابل دید قوی ناظر میله‌ای واقع در یک دامنه چندضلعی‌گونه و نیز ساده‌سازی چندضلعی قابل دید ضعیف ناظر میله‌ای واقع در یک چندضلعی ساده قابل استفاده است[۱۴۲، ۱۴۱].

ساده‌سازی (چندضلعی قابل دید) در حالت جویباری

در این رساله، یک الگوریتم کلی برای حل مساله ساده‌سازی در حالت جویباری[۱۲] نیز ارائه شده است [۲]. در این مساله، می‌خواهیم یک مسیر که ممکن است مرز دید یک ناظر باشد و نقاط آن بصورت جویباری و به ترتیب دیده می‌شوند را با انتخاب k نقطه از آنها تقریب بزنیم بگونه‌ای که خطای این تقریب با منحنی اولیه کمینه باشد. در این مساله فرض بر این است که تعداد نقاط ورودی بسیار زیاد است و همه آنها قابل ذخیره در حافظه نیستند.

نتایجی که در مورد این مساله برای مدل ورودی جویباری بدست آمده است برای معیارهای خطای فِرشه[۱۳] [۳]، هاوس‌دورف[۱۴] [۱] و خطای وابسته به دید [۱۴۵] به شرح زیرند:

- برای مسیر ورودی دلخواه و معیار خطای فاصله هاوس‌دورف هیچ الگوریتمی با حافظه کمتر از تعداد نقاط ورودی وجود ندارد که ضریب تقریب آن مقدار متناهی c باشد.
- با داشتن شرط $\mathcal{A}$ و رویه $\mathcal{O}$، رابطه $err(\mathcal{I}(k)) \leq c \cdot e \cdot err(Opt(\frac{k}{۴}))$ در مورد الگوریتم $\mathcal{I}$ وجود دارد که $err(\mathcal{I}(k))$ خطای تقریب مربوط به الگوریتم $\mathcal{I}$ با k نقطه و $err(Opt(\frac{k}{۴}))$ خطای مربوط به الگوریتم بهینه با $\frac{k}{۴}$ نقطه است. در اینصورت الگوریتم $\mathcal{I}$ را ce–رقابتی می‌گوییم.

---

[۱۱] Line Simplification

[۱۲] Streaming

[۱۳] Fréchet

[۱۴] Hausdorff

— شرط $\mathcal{A}$: $err(p_ip_m) \leq err(p_ip_j),\ i \leq l \leq m \leq j$ که $err(p_ip_j)$ خطای مربوط به بخشی از منحنی است که بین نقاط p_i و p_j قرار دارند و با پاره‌خط p_ip_j تقریب زده شده است.

— رویه $\mathcal{O}$: مقدار $err(p_ip_j)$ را می‌توان با ضریب تقریب ثابت e تعیین کرد.

— الگوریتم $\mathcal{I}$: با مشاهده یک نقطه جدید از بین $k+۳$ نقطه موجود، نقطه‌ای که کمترین خطا را ایجاد می‌کند حذف می‌شود.

- مواردی که شرط $\mathcal{A}$ برای آنها صدق می‌کند:

— معیار خطای هاوس‌دورف برای مسیرهای محدب: $c = ۱$

— معیار خطای هاوس‌دورف برای مسیرهای صعودی (xy-یکنوا): $c = ۲$

— معیار خطای فِرشه برای هر مسیر دلخواه: $c = ۲$

— معیار خطای وابسته به دید برای چندضلعی قابل دید ناظر نقطه‌ای: $c = ۲$

— معیار خطای وابسته به دید برای چندضلعی قابل دید قوی ناظر میله‌ای: $c = ۲$

- مواردی که رویه $\mathcal{O}$ برای آنها ارائه شده است:

— رویه $\mathcal{O}$ برای مسیرهای محدب و معیار خطای هاوس‌دورف با نگهداری مساحت بین مسیر و خط تقریب قابل تهیه است: $e = ۳$

— رویه $\mathcal{O}$ برای مسیرهای صعودی و معیار خطای هاوس‌دورف با نگهداری مجموعه هسته[۱۵] قابل تهیه است: $e = ۲ + \epsilon$

— رویه $\mathcal{O}$ برای مسیرهای دلخواه و معیار خطای فِرشه با نگهداری مجموعه هسته و مسیر برگشتی منحنی قابل تهیه است: $e = ۲\sqrt{۲} + \epsilon$

— رویه $\mathcal{O}$ برای چندضلعی قابل دید ناظر نقطه‌ای و معیار خطای فاصله‌ای وابسته به دید با نگهداری مجموعه هسته قابل تهیه است: $e = ۱ + \epsilon$

— رویه $\mathcal{O}$ برای چندضلعی قابل دید قوی ناظر میله‌ای و معیار خطای فاصله‌ای وابسته به دید با نگهداری مجموعه هسته قابل تهیه است: $e = ۱ + \epsilon$

- بنابراین الگوریتم $\mathcal{I}$ برای موارد زیر دارای ضریب رقابتی $O(۱)$ است:

— الگوریتم $\mathcal{I}$ برای مسیرهای محدب و با معیار خطای هاوس‌دورف، ۳-رقابتی است.

— الگوریتم $\mathcal{I}$ برای مسیرهای صعودی و با معیار خطای هاوس‌دورف، $(\epsilon + ۴)$-رقابتی است.

[۱۵] Core-Set

- الگوریتم $\mathcal{I}$ برای مسیرهای دلخواه و با معیار خطای فِرشه، $(\epsilon + 4\sqrt{2})$-رقابتی است.
- الگوریتم $\mathcal{I}$ برای چندضلعی قابل دید ناظر نقطه‌ای و با معیار خطای وابسته به دید، $(\epsilon + 2)$-رقابتی است.
- الگوریتم $\mathcal{I}$ برای چندضلعی قابل دید ناظر میله‌ای و با معیار خطای وابسته به دید، $(\epsilon + 2)$-رقابتی است.

نگهداری روابط دیداری در فضای سه‌بعدی

یکی از ایده‌های جالب و کاربردی برای حل مسائل قابلیت دید، داده‌ساختاری بنام مجتمع قابلیت دید است که شامل تمام روابط دیداری بین اشیای موجود در یک صحنه است. این داده‌ساختار برای حالت سه‌بعدی به علت پیچیدگی بالای آن عملا غیرکاربردی است. به عنوان بخشی از این کار تحقیقاتی تلاش شد تا با اصلاحاتی در این داده‌ساختار امکان استفاده از آن برای محاسبه دید فراهم شود [۱۰۱].

۱-۵ ساختار رساله

مطالب این رساله بصورت زیر تدوین شده‌اند. در فصل ۲، مفاهیم، تعاریف و داده‌ساختارهای استفاده شده در این رساله و نیز الگوریتم‌های پایه‌ای محاسبه دید در صفحه بیان می‌شود. در فصل ۳، ایده‌ی پیش‌پردازش برای بالا بردن سرعت پاسخ‌گویی در زمان پرس‌وجو بیان می‌شود و الگوریتم‌هایی که بر این اساس برای چندضلعی‌های ساده ارائه شده‌اند توضیح داده می‌شوند. در فصل ۴، نتایج بدست آمده برای تعیین دید یک ناظر نقطه‌ای در دامنه‌های چندضلعی‌گونه با استفاده از پیش‌پردازش بیان می‌شود. در فصل ۵، مساله نگهداری دید برای یک نقطه متحرک در دامنه‌های چندضلعی‌گونه بررسی و برای آن دو الگوریتم ارائه می‌شود. در فصل ٦، روش نگهداری دید از یک پاره‌خط متحرک در چندضلعی ساده ارائه می‌شود. در فصل ۷، مساله دید در فضای سه بعدی مطرح می‌شود و نسخه مناسبی از مجتمع قابلیت دید برای آن ارائه می‌شود.

در فصل ۸، مفهوم ساده‌سازی چندضلعی قابل دید بیان و برای آن یک معیار ساده‌سازی مبتنی بر دید ارائه می‌شود. همچنین، الگوریتم‌هایی برای ساده‌سازی چندضلعی قابل دید براساس این معیار ارائه می‌شود. در فصل ۹، ساده‌سازی در حالت جویباری مطالعه و برای آن یک الگوریتم عمومی ارائه می‌شود که هم در مورد چندضلعی قابل دید و هم با معیارهای معمول ساده‌سازی قابل استفاده است. در فصل ۱۰، جمع‌بندی مطالب ارائه شده در رساله و روش‌ها و مسائل قابل توسعه در ادامه این رساله بیان می‌شوند.

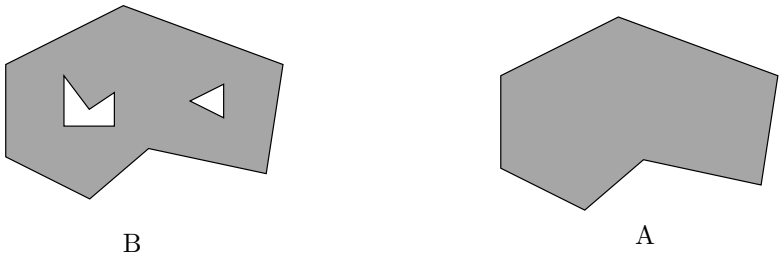
فصل ۲

مفاهیم، داده‌ساختارها و الگوریتم‌های پایه

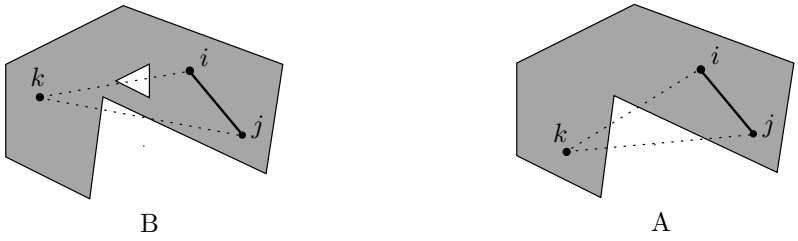
در این فصل مفاهیم عمومی بکاررفته در این رساله که شامل تعریف قابلیت دید و محیط‌های مرجع است بصورت رسمی تعریف می‌شوند. علاوه‌برآن، الگوریتم‌های پایه‌ای مربوط به قابلیت دید که در فصل‌های بعدی مورد استفاده قرار گرفته‌اند ارائه می‌شوند. به همین منظور، پس از ارائه تعاریف در بخش ۲—۱، الگوریتم‌های محاسبه فضای قابل دید از یک ناظر نقطه‌ای واقع در یک چندضلعی ساده و حفره‌دار به ترتیب در بخش‌های ۲—۲ و ۲—۳ بیان می‌شوند. در بخش ۲—۴، قابلیت دید از یک پاره‌خط بررسی و در بخش ۲—۵ الگوریتم محاسبه گراف قابلیت دید بیان خواهد شد.

۲-۱ تعاریف و مفاهیم قابلیت دید

یک چندضلعی $\mathcal{P}$ در یک صفحه که با رئوس v_1، v_2، ... v_n مشخص شده است شامل همه نقاط صفحه است که توسط مرز چندضلعی محدود شده‌اند. مرز چندضلعی $\mathcal{P}$ شامل زنجیره پاره‌خط‌های v_iv_{i+1} (برای $1 \leq i < n$) و نیز v_nv_1 است. چندضلعی $\mathcal{P}$ ساده است هرگاه پاره‌خط‌های مرز چندضلعی فقط در نقاط انتهایی همدیگر را قطع کنند و هیچ دو نقطه‌ای از مجموعه نقاط v_i همپوشانی نداشته باشند. نقاط v_i رئوس چندضلعی $\mathcal{P}$ را تشکیل می‌دهند. چندضلعی $\mathcal{P}$ حفره‌دار نامیده می‌شود هرگاه مرز بیرونی آن یک چندضلعی ساده باشد و علاوه بر آن تعدادی چندضلعی ساده که با هم و با مرز بیرونی تقاطع ندارند از داخل چندضلعی بیرونی کنده شده باشند. هر یک از این چندضلعی‌ها یک حفره در چندضلعی بیرونی ایجاد می‌کند. در شکل ۲—۱ یک چندضلعی ساده و یک چندضلعی حفره‌دار نشان داده شده است.



شکل ۲–۱: A) چندضلعی ساده، B) چندضلعی حفره‌دار.

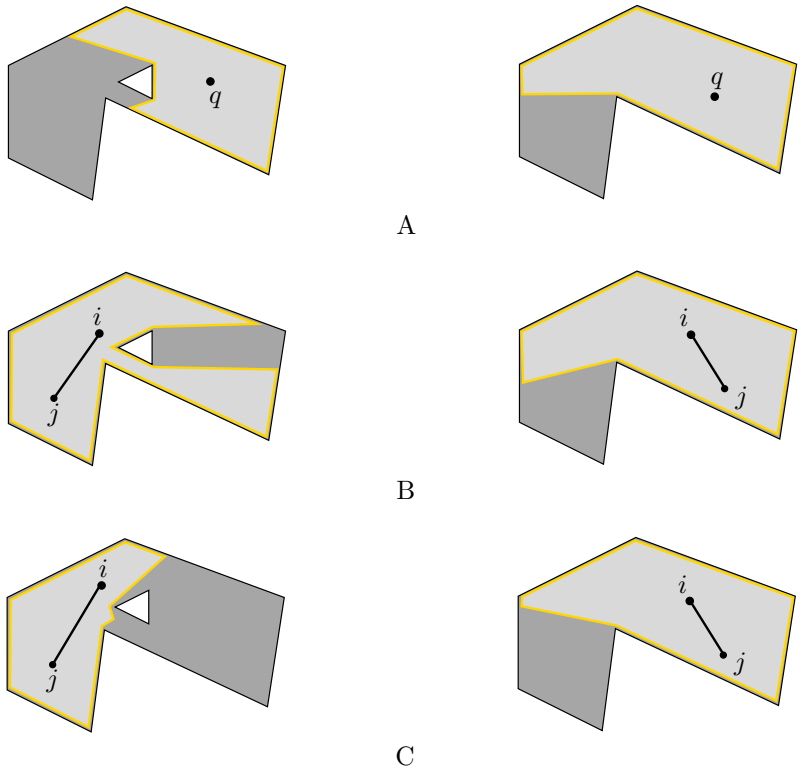


شکل ۲–۲: A) قابلیت دید در چندضلعی ساده، B) قابلیت دید در چندضلعی حفره‌دار.

هر چندضلعی ساده $\mathcal{P}$ با دنباله رئوس آن و هر چندضلعی حفره‌دار با چندضلعی ساده بیرونی و چندضلعی‌های ساده داخلی آن مشخص می‌شود. تعداد رئوس هر چندضلعی برابر با تعداد رئوس مرز بیرونی و نیز حفره‌های داخل آن است. پاره‌خط‌های متصل کننده رئوس مجاور نیز اضلاع چندضلعی را تشکیل می‌دهند.

دو نقطه در داخل یک چندضلعی همدیگر را می‌بینند یا از یکدیگر قابل دید هستند هرگاه پاره‌خط متصل کننده آنها بطور کامل درون چندضلعی قرار بگیرد و از بیرون چندضلعی یا از درون حفره‌های آن عبور نکند. در شکل ۲—۲ نقاط i و j از همدیگر قابل دید هستند در حالیکه نقاط i و k و نیز j و k از همدیگر قابل دید نیستند.

مجموعه‌ی همه نقاطی که از نقطه‌ی q واقع در درون چندضلعی $\mathcal{P}$ قابل دید هستند چندضلعی قابل دید q نامیده و با $V(q)$ نشان داده می‌شود. چندضلعی قابل دید از یک نقطه برای چندضلعی‌های ساده و حفره‌دار در قسمت A از شکل ۲—۳ نشان داده شده است. برای یک پاره‌خط که بطور کامل درون چندضلعی $\mathcal{P}$ قرار دارد دو نوع چندضلعی قابل دید تعریف می‌شود: چندضلعی قابل دید ضعیف برای پاره خط ij واقع در درون $\mathcal{P}$ شامل همه نقاطی از $\mathcal{P}$ است که از حداقل یک نقطه‌ی این پاره‌خط قابل دید باشند. این نوع از قابلیت دید در قسمت B از شکل ۲—۳ نشان داده شده است. نوع دیگر قابلیت دید برای پاره خط ij که قابلیت دید قوی نامیده می‌شود شامل همه نقاطی از $\mathcal{P}$ است که از هر یک از نقاط این پاره‌خط قابل دید باشند. در قسمت C از شکل ۲—۳ این نوع از قابلیت دید نشان داده شده است. قابلیت دید برای یک چندضلعی داخل $\mathcal{P}$ نیز بصورت مشابه قابل تعریف است.



شکل ۲-۳: چندضلعی قابل دید در چندضلعی‌های ساده و حفره‌دار: A) چندضلعی قابل دید از نقطه q، B) چندضلعی قابل دید ضعیف از پاره‌خط ij، C) چندضلعی قابل دید قوی از پاره خط ij.

همه الگوریتم‌های مربوط به محاسبه چندضلعی قابل دید برای پاره‌خط یا چندضلعی‌های درون $\mathcal{P}$ مبتنی بر محاسبه چندضلعی قابل دید نقطه‌ای هستند. به عنوان مثال، قابلیت دید قوی در چندضلعی‌های ساده از اشتراک چندضلعی‌های قابل دید نقاط دو سر پاره‌خط بدست می‌آید. محاسبه قابلیت دید ضعیف نیز با حرکت کردن از یک انتهای پاره‌خط به انتهای دیگر و توسعه‌ی چندضلعی قابل دید نقطه‌ای در طول مسیر بدست می‌آید.

بنابراین مساله کلیدی در حل مسائل قابلیت دید در صفحه، محاسبه‌ی قابلیت دید نقطه‌ای یا $V(q)$ است که در بخش‌های بعدی این فصل به آن می‌پردازیم.

بدیهی است که با داشتن رئوس $V(q)$ می‌توان آن‌را بطور دقیق مشخص کرد. علاوه‌برآن، داشتن دنباله رئوس یا یال‌هایی از $\mathcal{P}$ که توسط q دیده می‌شوند برای مشخص کردن دقیق $V(q)$ کافی است. برای این‌کار دنباله رئوس $\mathcal{P}$ در تناظر با دنباله مزبور پیمایش می‌شوند و $V(q)$ که ممکن است در برخی موارد فقط شامل بخشی از یک ضلع از $\mathcal{P}$ باشد بطور دقیق مشخص می‌شود. به همین جهت در هنگام محاسبه $V(q)$ یا نمایش آن اصراری بر دقیق بودن $V(q)$ نداریم و فقط تعیین دنباله رئوس یا یال‌هایی که بطور کامل یا بخش‌هایی از آنها از q قابل دید هستند مد نظر است.

گراف قابلیت دید G برای مجموعه‌ی نقاط p_1، p_2، ...، p_n واقع در یک چندضلعی به این ترتیب تعریف می‌شود که رئوس G همان نقاط p_i است و بین دو راس یال است هرگاه پاره‌خط بین نقاط متناظر با آن رئوس بطور کامل درون چندضلعی قرار بگیرد.

۲-۲ چندضلعی قابل دید نقطه‌ای در چندضلعی‌های ساده

برای تعیین $V(q)$ حداقل یک‌بار باید همه رئوس و اضلاع $\mathcal{P}$ پیمایش شوند. بنابراین برای چندضلعی n راسی $\mathcal{P}$ محاسبه $V(q)$ از مرتبه $\Omega(n)$ است. نخستین الگوریتم بهینه برای تعیین $V(q)$ در سال ۱۹۸۱ ارائه شد [۵۴]. پس از آن الگوریتم‌های بهینه دیگری نیز برای حل این مساله ارائه شدند [۹۰، ۳۱، ۶۲]. در این جا شهود کلی الگوریتم اول بیان می‌شود.

برای محاسبه $V(q)$ برای نقطه q که درون چندضلعی ساده $\mathcal{P}$ مطابق قسمت A از شکل ۲—۴ قرار دارد، از یک نقطه دلخواه مرز $\mathcal{P}$ که از q قابل دید است شروع می‌کنیم و در جهت عقربه‌های ساعت مرز $\mathcal{P}$ را پیمایش می‌کنیم تا بتدریج $V(q)$ ساخته شود. در این شکل نقطه شروع محل تقاطع خط افقی گذرنده از q با مرز $\mathcal{P}$ است که با STR مشخص شده است.

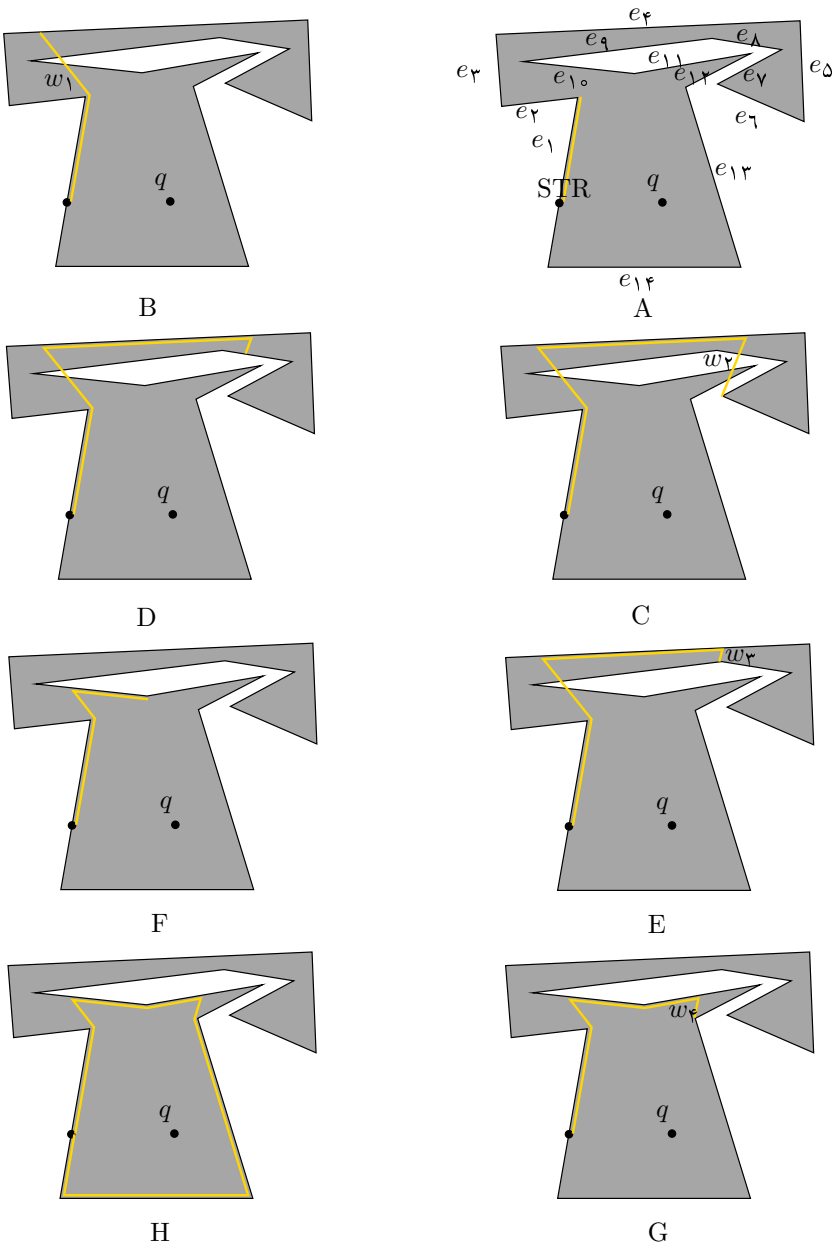
از نقطه شروع و در جهت عقربه‌های ساعت رئوس و اضلاع چندضلعی پیموده می‌شوند و برای هر ضلع تعیین می‌شود که آیا این ضلع با در نظر گرفتن اضلاع پیموده شده قبلی از نقطه q قابل دید است یا خیر. برای پاسخ به این سوال حالات مختلفی را باید بررسی کرد که در شکل ۲—۴ نشان داده شده‌اند.

در قسمت A ضلع e_1 از q قابل دید است. پس آن‌را به مجموعه اضلاع قابل دید اضافه می‌کنیم. اضلاع قابل دید به ترتیب دیده شدن در یک پشته نگهداری می‌شوند تا بهنگام‌سازی آن‌ها سریع‌تر انجام شود.

چنانچه در قسمت B نشان داده شده است، ضلع e_2 به علت زاویه آن با ضلع قبلی نسبت به نقطه q، از این نقطه قابل دید نیست. به همین دلیل امتداد دید نقطه q از راس انتهایی e_1 که یک پنجره نامیده می‌شود و با w_1 نشان داده شده است به عنوان ضلع قابل دید به پشته اضافه می‌شود.

در چنین مواردی که یک پنجره در بالای پشته قرار می‌گیرد، هنگام پردازش اضلاع چندضلعی باید بررسی شود که آیا این اضلاع در سمت قابل دید یا در سمت غیر قابل دید این پنجره قرار می‌گیرند و چنانچه همانند ضلع e_3 بطور کامل در قسمت غیرقابل دید باشند تغییری در پشته ایجاد نمی‌شود ولی اگر همانند ضلع e_4 بخشی از آنها در قسمت قابل دید باشد، با تعیین اندازه پنجره و نیز افزودن آن ضلع، پشته بهنگام می‌شود. ضلع e_5 نیز به همین ترتیب به پشته اضافه می‌شود.

هنگام پردازش ضلع e_6 مشاهده می‌شود که با توجه به زاویه آن با ضلع قبلی و وضعیت آن نسبت به نقطه q قابل دید نیست و باعث پوشیده شدن بخش‌هایی که قبلا قابل دید بودند نیز می‌شود. در این حالت همانند قسمت C از شکل ۲—۴ پنجره w_2 باید به بالای پشته اضافه شود و



شکل ۲–۴: محاسبه چندضلعی قابل دید نقطه‌ای در یک چندضلعی ساده.

قبل از آن نیز همه اضلاعی که قبلا قابل دید بودند و اکنون پشت این پنجره نسبت به نقطه q قرار دارند باید از پشته حذف شوند.

پردازش ضلع $e_۷$ نیز چنانچه در قسمت D از شکل ۲–۴ آمده است، باعث تغییر اندازه پنجره $w_۲$ در پشته می‌شود. پردازش ضلع $e_۸$ نیز باعث حذف شدن پنجره $w_۲$ از پشته و ظاهر شدن پنجره $w_۳$ می‌شود که این وضعیت در قسمت E از شکل ۲–۴ نشان داده شده است.

به همین ترتیب ضلع $e_۹$ باعث می‌شود که همه اضلاع قابل دید از q و واقع در پشته تا پنجره $w_۱$ غیر قابل دید شوند. پردازش ضلع $e_{۱۰}$ نیز باعث تغییر اندازه پنجره $w_۱$ و نیز ظاهر شدن یک ضلع جدید در پشته می‌شود که در قسمت F از شکل ۲–۴ نشان داده شده است.

پردازش $e_{۱۱}$ باعث اضافه شدن این ضلع به پشته می‌شود که در قسمت G از شکل ۲–۴ نشان داده شده است. ضلع $e_{۱۲}$ باعث غیر قابل دید شدن بخشی از ضلع $e_{۱۱}$ و ظاهر شدن پنجره $w_۴$ در بالای پشته می‌شود. مطابق قسمت H از شکل ۲–۴ اضلاع $e_{۱۳}$ و $e_{۱۴}$ نیز به علت قابل دید بودن از q به پشته اضافه می‌شوند. پس در پایان و پس از رسیدن به نقطه شروع همه دنباله اضلاع و نیز پنجره‌های قابل دید از q در پشته و به ترتیب قرار دارند. این دنباله چندضلعی قابل دید از q را تشکیل می‌دهد.

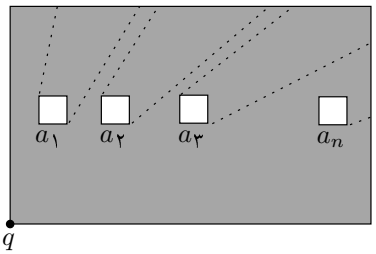
قضیه ۱ . چندضلعی قابل دید نقطه q واقع در درون چندضلعی ساده و n راسی $\mathcal{P}$ در زمان $O(n)$ قابل محاسبه است.

اثبات. در الگوریتم قبل با توجه به پردازش ترتیبی همه اضلاع $\mathcal{P}$، در هر لحظه دنباله رئوس قابل دید نسبت به دنباله پردازش شده محاسبه شده است. پس از پایان الگوریتم و رسیدن به نقطه شروع، همه اضلاع پردازش می‌شوند و در نتیجه اضلاع قابل دید از نقطه q نسبت به کل اضلاع $\mathcal{P}$ محاسبه شده‌اند که معادل با چندضلعی قابل دید q در $\mathcal{P}$ است.

زمان اجرای این الگوریتم نیز نسبت به تعداد اضلاع $\mathcal{P}$ خطی است زیرا هر یک از اضلاع $\mathcal{P}$ فقط یک‌بار پردازش می‌شوند. هنگام پردازش هر ضلع نیز فقط وضعیت آن ضلع نسبت به ضلع بالای پشته بررسی می‌شود و در صورت غیر قابل دید شدن ضلع بالای پشته آن ضلع از پشته حذف می‌شود و همین عمل برای ضلع جدیدی که در بالای پشته قرار می‌گیرد تکرار می‌شود. چون هر ضلع فقط یک‌بار به پشته اضافه یا از آن حذف می‌شود تعداد عملیات حذف و افزودن به پشته نیز $O(n)$ است. بنابراین زمان اجرای کل الگوریتم نسبت به تعداد رئوس $\mathcal{P}$ خطی است. □

۲–۳ چندضلعی قابل دید نقطه‌ای در چندضلعی‌های حفره‌دار

الگوریتمی که در بخش ۲–۲ برای چندضلعی‌های ساده بیان شد در مورد چندضلعی‌های حفره‌دار قابل استفاده نیست. این مساله به خاطر عدم وجود ترتیب یکسان و خوش تعریفی برای اضلاع این



شکل ۲–۵: مساله مرتب‌سازی قابل کاهش به مساله تعیین چندضلعی قابل دید نقطه‌ای در چندضلعی‌های حفره‌دار است و در نتیجه کران پایین این مساله $O(n \log n)$ است.

نوع چندضلعی‌ها است. علاوه‌برآن، حل این مساله برای چندضلعی‌های با مانع از پیچیدگی زمانی بیشتری برخوردار است.

قضیه ۲ . حد پایین مرتبه زمانی الگوریتم‌های تعیین چندضلعی قابل دید یک نقطه از یک چندضلعی حفره‌دار n راسی $O(n \log n)$ است.

اثبات. برای اثبات این قضیه مساله مرتب سازی را به این مساله تبدیل می‌کنیم. فرض کنید که اعداد a_1 تا a_n وجود دارند و می‌خواهیم آن‌ها را بصورت افزایشی مرتب کنیم. مطابق شکل ۲–۵ به ازای هر یک از این اعداد یک مربع در نظر گرفته می‌شود که همه این مربع‌ها در امتداد یک خط افقی و متناسب با ترتیب اعداد مکان‌دهی شده‌اند. این مربع‌ها به عنوان حفره‌هایی در یک مستطیل بزرگ که همه آنها را در بر گرفته است تصور می‌شوند. بدیهی است که تعداد رئوس این چندضلعی حفره‌دار $4n + 4$ یا $O(n)$ است. حال اگر چندضلعی قابل دید از نقطه q واقع در گوشه پایین–سمت چپ از مستطیل بزرگ را محاسبه کنیم، در این چندضلعی همه مربع‌ها به ترتیب اعداد متناظر با آنها در $V(q)$ ظاهر می‌شوند.

بنابراین اگر چندضلعی قابل دید از q را داشته باشیم با پیمایش آن ترتیب افزایشی اعداد a_1 تا a_n بدست می‌آید. از آنجا که کران پایین الگوریتم‌های مرتب سازی $O(n \log n)$ است پس کران پایین الگوریتم‌های تعیین چندضلعی قابل دید یک نقطه در چندضلعی‌های حفره‌دار نیز $O(n \log n)$ است. □

با توجه به کران پایین مساله، الگوریتم بهینه با مرتبه زمانی $O(n \log n)$ کاملا سرراست است. با یک خط جاروب شعاعی حول نقطه دید دنباله‌ی اضلاع و رئوس قابل دید مشخص می‌شود. در این روش همه رئوس بر اساس مختصات قطبی آنها مرتب می‌شوند و سپس از مبدا این مختصات یعنی محور x و با یک خط جاروب حول نقطه دید اضلاع چندضلعی پردازش می‌شوند. نقاط رویداد همان رئوس چندضلعی است و در هر لحظه امتداد خط جاروب تعدادی از اضلاع چندضلعی را قطع می‌کند که همواره نزدیک‌ترین آنها به q از این نقطه قابل دید است. این مجموعه از پاره‌خط‌ها در یک درخت جستجوی دودویی متوازن نگهداری می‌شوند. در هر نقطه رویداد

تعدادی پاره‌خط به این درخت افزوده یا از آن حذف می‌شود. پس از پایان جاروب، دنباله اضلاع قابل دید از نقطه q بدست می‌آید.

قضیه ۳ . چندضلعی قابل دید از هر نقطه از داخل یک چندضلعی حفره‌دار n راسی در زمان $O(n \log n)$ قابل محاسبه است.

اثبات. در الگوریتم بالا تعداد نقاط رویداد $O(n)$ است. در هر رویداد ممکن است تعدادی عنصر جدید به درخت جستجوی دودویی متوازن افزوده یا از آن حذف شود. با این حال تعداد کل عملیات افزودن و حذف کردن برابر با تعداد اضلاع چندضلعی یعنی $O(n)$ است. برای هر یک از این عملیات نیز $O(\log n)$ زمان لازم است. بنابراین کل زمان اجرا $O(n \log n)$ است. □

نخستین الگوریتم‌های حل این مساله با زمان اجرای $O(n \log n)$ در [۱۳، ۱۲۴] ارائه شدند که بعدا زمان اجرای حل این مساله به مقدار بهینه $O(n + h \log h)$ در [۷۱] بهبود یافت.

۲-۴ چندضلعی قابل دید از یک پاره‌خط

همانطوریکه در بخش ۲—۱ گفته شد، دو نوع چندضلعی قابل دید برای یک پاره‌خط تعریف می‌شود: چندضلعی قابل دید قوی و ضعیف. در یک چندضلعی ساده هر گاه یک نقطه از دو سر یک پاره‌خط واقع در درون آن چندضلعی قابل دید باشد آنگاه از همه نقاط آن پاره‌خط قابل دید است. برعکس این مطلب نیز بطور بدیهی صادق است. بنابراین چندضلعی قابل دید قوی برای یک پاره‌خط واقع در درون یک چندضلعی ساده از اشتراک چندضلعی‌های قابل دید دو سر آن پاره‌خط بدست می‌آید.

چندضلعی قابل دید نقاط دو سر پاره‌خط در $O(n)$ و محاسبه اشتراک این دو چندضلعی نیز در همین زمان انجام‌پذیر است که با یک پیمایش ترتیبی بر روی دو چندضلعی بدست می‌آید. بنابراین تعیین چندضلعی قابل دید قوی در چندضلعی‌های ساده در زمان خطی امکان‌پذیر است.

در مورد چندضلعی قابل دید ضعیف یک پاره‌خط واقع در داخل یک چندضلعی ساده ابتدا یک الگوریتم با مرتبه زمانی $O(n \log n)$ ارائه شد [۳۱]. سپس یک الگوریتم بهینه خطی برای این مساله ارائه شد که مبتنی بر مساله کوتاه‌ترین مسیر و نیز مثلث‌بندی چندضلعی‌های ساده است [۶۲].

حل این مساله در چندضلعی‌های حفره‌دار به مراتب پیچیده‌تر است. در [۱۲۴] اثبات شده است که محاسبه چندضلعی قابل دید یک پاره‌خط واقع در یک چندضلعی حفره‌دار n راسی $\Omega(n^4)$ است و یک الگوریتم بهینه برای آن ارائه شده است. این الگوریتم مبتنی بر گراف قابلیت دید رئوس چندضلعی است.

برای محاسبه چندضلعی قابل دید از یک چندضلعی واقع در درون $\mathcal{P}$ نیز می‌توان از چندضلعی‌های قابل دید مربوط به اضلاع آن استفاده کرد که از اجتماع آنها بدست می‌آید و به دلیل عدم کاربرد و پیچیدگی بالا کمتر به آن پرداخته شده است.

۲-۵ محاسبه گراف قابلیت دید دو بعدی

گراف قابلیت دید یکی از داده‌ساختارهای مناسب برای حل مسائل قابلیت دید است. در این گراف به ازای هر یک از رئوس چندضلعی یک راس وجود دارد و دو راس گراف به هم یال دارند هرگاه رئوس متناظر آنها در چندضلعی نسبت به هم قابل دید باشند. واضح است که اندازه این گراف برای یک چندضلعی(ساده یا حفره‌دار) n راسی $\Omega(n)$ و $O(n^2)$ است.

با توجه به کاربردهای متعدد این گراف در مسائل قابلیت دید، برنامه‌ریزی حرکت، تعیین کوتاه‌ترین مسیر و مسائل نگهبانی، الگوریتم‌های کارایی برای محاسبه آن ارائه شده است. اولین الگوریتم غیربدیهی توسط لی ارائه شد که زمان اجرای آن $O(n^2 \log n)$ است [۸۹]. سپس مجموعه‌ای از الگوریتم‌های با زمان اجرای $O(n^2)$ ارائه شدند که مقدار حافظه مصرفی آنها $O(n^2)$ [۱۴، ۱۳۳] یا $O(n)$ [۴۳، ۱۰۶] است. همچنین، چندین الگوریتم کارا با زمان اجرای وابسته به خروجی ارائه شد که دارای زمان اجرای $O(e \log n)$ [۱۰۶] یا $O(e + n \log n)$ [۵۰، ۵۱، ۸۲، ۸۳، ۱۰۸، ۱۱۵] هستند که e تعداد یال‌های گراف قابلیت دید است.

برای محاسبه گراف قابلیت دید در زمان $O(n^2 \log n)$ کافی است $V(p)$ را به ازای هر یک از رئوس چندضلعی مرجع محاسبه کنیم. این کار در زمان $O(n \log n)$ قابل انجام است و با داشتن $V(p)$ یالهای مجاور راس متناظر با p در گراف قابلیت دید در دست است.

فصل ۳

پیش‌پردازش برای بهبود سرعت پاسخ‌گویی به مسائل قابلیت دید

الگوریتم‌هایی که در فصل ۲ ارائه شد به صورت بهینه مساله چندضلعی قابل دید نقطه‌ای را در چندضلعی‌های ساده و حفره‌دار حل می‌کنند. این الگوریتم‌ها در واقع در بدترین حالت بهینه هستند. ولی واقعیت این است که در اغلب موارد اندازه چندضلعی قابل دید به مراتب کوچک‌تر از پیچیدگی چندضلعی مرجع است. از طرف دیگر اگر تعداد نقاط پرس‌وجو زیاد باشد اجرای الگوریتم‌های یاد شده در این شرایط از کارایی مناسب برخوردار نیست.

برای رسیدن به دو هدف مطالعه برای یافتن الگوریتم‌های دیگری شروع شد. هدف اول یافتن الگوریتم‌هایی بود که کارایی آنها متناسب با اندازه خروجی مساله باشد. هدف دوم پاسخ‌گویی به تعداد پرس‌وجوی زیاد است. برای برآوردن این اهداف لازم بود که با انجام پیش‌پردازش بر روی چندضلعی مرجع، اطلاعاتی از وضعیت هندسی آن جمع‌آوری و نگهداری شود. این اطلاعات در زمان پرس‌وجو مورد استفاده قرار گرفته و باعث افزایش سرعت در جواب‌گویی به پرس‌وجوها می‌شود.

نخستین بار در [۱۵] الگوریتمی ارائه شد که برای چندضلعی‌های حفره‌دار با زمان پیش پردازش $O(n^2)$ و مصرف همین اندازه حافظه می‌توانست $V(q)$ را در زمان $O(n)$ محاسبه کند. مشاهده می‌شود که این الگوریتم با وجود اینکه زمان پرس‌وجوی آن متناسب با اندازه خروجی یعنی $V(q)$ نیست ولی از الگوریتم بهینه‌ای که در فصل قبل ارائه شد مناسب‌تر است. نخستین الگوریتم با زمان اجرای متناسب با $V(q)$ در [۱۳۱] ارائه شد. این الگوریتم با زمان پیش پردازش و حافظه مصرفی $O(n^2)$ دارای زمان پاسخ $O(|V(q)|\log(n/|V(q)|))$ است. الگوریتم دیگری نیز در [۷۵] ارائه شده است که مبتنی بر ایجاد یک داده‌ساختار با اندازه خطی است که بر اساس آن در زمان $O(\log n)$ می‌توان مساله امتداد اشعه را حل کرد. با این روش می‌توان با زمان پیش پردازش و حافظه

مصرفی خطی، $V(q)$ را در زمان $O(|V(q)|\log n)$ برای هر نقطه پرس‌وجوی q محاسبه کرد.

در این الگوریتم‌ها نیز ضریب $\log n$ در زمان پرس‌وجو عامل محدود کننده است. نهایتا در [۶۴] و [۲۴] الگوریتم‌هایی با زمان پرس‌وجوی بهینه برای چندضلعی‌های ساده ارائه شد. زمان پرس‌وجو در این الگوریتم‌ها $O(|V(q)|+\log n)$ است که برای دست‌یابی به آن نیازمند پیش پردازشی با هزینه زمانی $O(n^۳\log n)$ و حافظه مصرفی $O(n^۳)$ هستند. این الگوریتم‌ها مبتنی بر افراز چندضلعی مرجع به ناحیه‌های قابلیت دید هستند که همه نقاط هر یک از این ناحیه‌ها دارای وضعیت دید هم‌ارزی هستند. باتوجه به زمان پیش‌پردازش بالا و نیز مصرف بالای حافظه‌ی این الگوریتم‌ها، فعالیت‌هایی برای کاهش هزینه پیش‌پردازش در [۱۱] صورت گرفت که منجر به یک الگوریتم با زمان پیش‌پردازش $O(n^۲\log n)$ و مصرف $O(n^۲)$ حافظه شد. در این الگوریتم $V(q)$ برای هر نقطه پرس‌وجوی دلخواه واقع در داخل چندضلعی ساده مرجع در زمان $O(|V(q)|+\log^۲ n)$ قابل محاسبه است.

یکی دیگر از روش‌های انجام پیش‌پردازش، مجتمع قابلیت دید است که در [۱۰۹] ارائه شد. بر اساس این روش با زمان پیش‌پردازش $O(n\log n+k)$ چندضلعی قابل دید هر نقطه پرس‌وجوی واقع در یک چندضلعی حفره‌دار را می‌توان در زمان $O(|V(q)|\log n)$ محاسبه کرد. در این الگوریتم k متناسب با اندازه گراف مماس قابلیت دید[۱] چندضلعی مرجع است. این داده‌ساختار در فصل ۷ توضیح داده می‌شود.

در ادامه این فصل ابتدا مفهوم افراز مبتنی بر قابلیت دید و سپس دو الگوریتم اصلی که بر اساس این مفهوم ارائه شده‌اند معرفی خواهند شد.

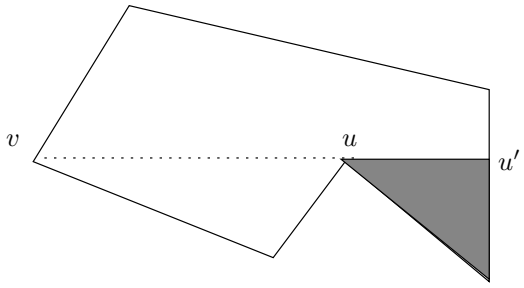
۳-۱ افراز مبتنی بر قابلیت دید

در چندضلعی ساده $\mathcal{P}$ ناحیه‌ی قابلیت دید R شامل یک مجموعه بیشینه از نقاط به هم پیوسته $\mathcal{P}$ است که هر دو نقطه از این ناحیه دنباله یکسانی از رئوس $\mathcal{P}$ را می‌بینند. به عبارت دیگر، چندضلعی قابل دید همه نقاط یک ناحیه قابلیت دید از رئوس و اضلاع یکسانی از $\mathcal{P}$ تشکیل شده است.

چنانچه همه ناحیه‌های قابلیت دید $\mathcal{P}$ شناسایی و مشخص شوند و برای هر ناحیه، دنباله رئوس قابل دید از آن تعیین شود، در زمان پرس‌وجو کافیست تا ناحیه قابلیت دیدی که نقطه پرس‌وجوی q درون آن قرار دارد مشخص شود. در این صورت دنباله رئوس مربوط به $V(q)$ که برابر با مجموعه رئوس متناظر با آن ناحیه است مهیا است و از روی آنها $V(q)$ ساخته می‌شود.

افراز مبتنی بر قابلیت دید روشی برای شناسایی ناحیه‌های قابلیت دید است. برای اینکه ناحیه‌های قابلیت دید مشخص شوند باید مرزهای این ناحیه‌ها و در واقع شرایطی که باعث تفاوت دید در دو نقطه می‌شود مشخص شود. چنانچه در شکل ۳—۱ نشان داده شده است، هر جا که یک

[۱] Tangent Visibility Graph



شکل ۳-۱: uu' یک پنجره از چندضلعی است.

راس انعکاسی[۲] وجود دارد باعث تفاوت دید در برخی از نقاط چندضلعی می‌شود. در این شکل نقاطی که در زیر پاره‌خط uu' قرار دارند راس v را نمی‌بینند در حالیکه نقاط بالای این پاره‌خط می‌توانند راس v را ببینند.

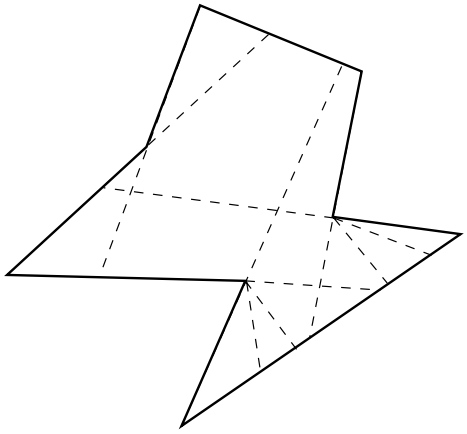
بنابراین پاره خط uu' جزء پاره‌خط‌های تعیین کننده مرز ناحیه‌های قابلیت دید است. این نوع پاره‌خط‌ها پنجره نامیده می‌شوند. هر دو راس u و v که همدیگر را می‌بینند و u یک راس انعکاس است، یک پنجره در چندضلعی بوجود می‌آورند که از امتداد دادن پاره‌خط vu از طرف u تا رسیدن به مرز چندضلعی بدست می‌آید.

براحتی قابل بررسی است که اضلاع چندضلعی همراه با پنجره‌های آن مرزهای ناحیه‌های قابلیت دید را تشکیل می‌دهند. بنابراین، برای تعیین ناحیه‌های قابلیت دید، همه پنجره‌های $\mathcal{P}$ مشخص می‌شوند. تفکیک حاصل از این پاره‌خط‌ها و اضلاع چندضلعی، ناحیه‌های قابلیت دید را مشخص می‌کند. این روش افراز مبتنی بر قابلیت دید گفته می‌شود. در شکل ۳—۲ افراز مبتنی بر قابلیت دید یک چندضلعی ساده نشان داده شده است.

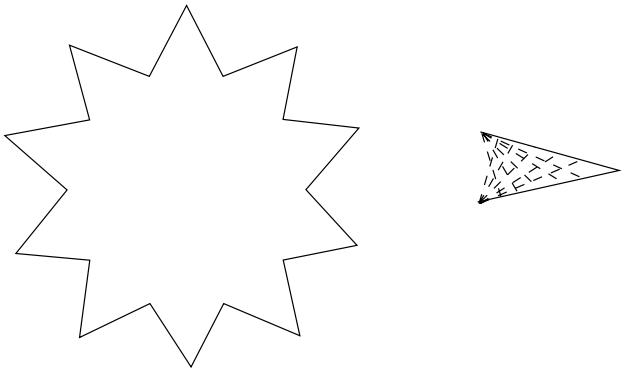
در صورت داشتن ناحیه‌های قابلیت دید و دنباله رئوس قابل دید از نقاط هر ناحیه، $V(q)$ را می‌توان به ازای هر نقطه پرس‌وجوی q محاسبه کرد. برای این کار ابتدا به عنوان پیش‌پرازش یک داده‌ساختار مکان‌یابی بر روی ناحیه‌های قابلیت دید ایجاد می‌کنیم. سپس در زمان پرس‌وجو با استفاده از داده‌ساختار مکان‌یابی، ناحیه در برگیرنده نقطه پرس‌وجو پیدا می‌شود. دنباله رئوس قابل دید از نقطه پرس‌وجو برابر با رئوس قابل دید مربوط به این ناحیه قابلیت دید است. با یک‌بار پیمایش این دنباله، رئوس و اضلاع $V(q)$ بطور دقیق مشخص می‌شوند.

با توجه به اینکه تعداد پنجره‌ها $O(n^2)$ است، چنانچه این پاره‌خط‌ها در $O(n^4)$ نقطه تلاقی داشته باشند تعداد ناحیه‌های قابلیت دید $O(n^4)$ خواهد شد که نتیجه‌ای بدیهی از فرمول اویلر برای گراف‌های مسطح است. از طرفی تعداد رئوس قابل دید از هر یک از این ناحیه‌ها $O(n)$ است. بنابراین هزینه انجام این پیش‌پردازش از مرتبه زمانی $O(n^4)$ و حافظه مورد نیاز برای نگهداری رئوس قابل دید برای همه ناحیه‌های قابلیت دید $O(n^5)$ خواهد شد. تهیه ساختار مکان‌یابی در

[۲] Reflex Vertex



شکل ۳-۲: افراز مبتنی بر قابلیت دید چندضلعی ساده.



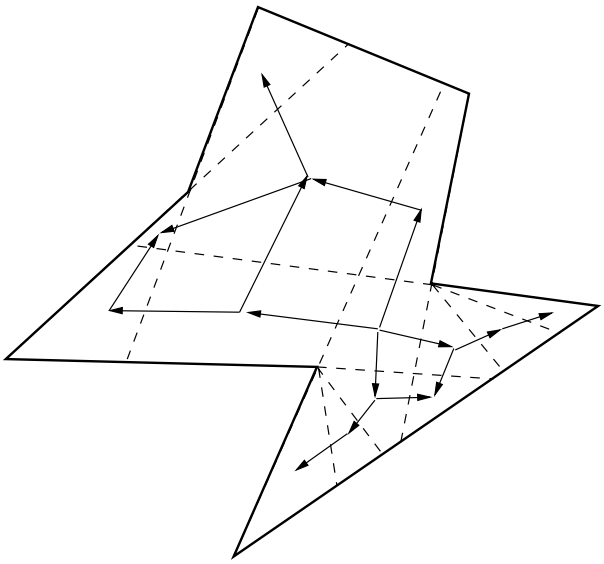
شکل ۳-۳: یک چندضلعی ساده با $O(n^3)$ ناحیه قابلیت دید.

زمان $O(n^4 \log n)$ و با حافظه $O(n^4)$ میسر است. با این مقدار پیش‌پردازش، زمان پرس‌وجو $O(|V(q)| + \log n)$ خواهد شد.

این تحلیل بسیار ابتدایی و خام است. در [۶۴] و [۲۴] نشان داده شده است که در یک چندضلعی ساده تعداد ناحیه‌های قابلیت دید $O(n^3)$ است. در شکل ۳—۳ نیز یک چندضلعی نشان داده شده است که دارای $O(n^3)$ ناحیه قابلیت دید است.

این نتیجه هزینه پیش پردازش را به صورت خطی کاهش می‌دهد. با این حال هنوز هم می‌توان هزینه پیش پردازش را کمتر کرد: دو ناحیه قابلیت دید مجاور که یک پنجره بین آنها قرار گرفته است رئوس یکسانی از $\mathcal{P}$ را می‌بینند مگر یک راس که مبدا پنجره بین آنها است. نقاط یکی از این ناحیه‌ها این راس را می‌بینند ولی نقاط ناحیه دیگر این راس را نمی‌بینند.

بنابراین نیازی به نگهداری رئوس قابل دید از همه ناحیه‌های قابلیت دید نیست و می‌توان از اطلاعات پنجره‌ها و نیز رئوس قابل دید برخی از ناحیه‌ها، رئوس قابل دید سایر ناحیه‌ها را تعیین



شکل ۳-۴: گراف دوگان جهت‌دار متناظر با افراز مبتنی بر قابلیت دید.

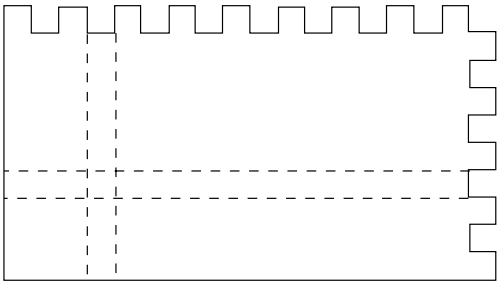
کرد. برای این منظور گراف دوگان جهت‌دار مربوط به افراز مبتنی بر قابلیت دید ساخته می‌شود. در این گراف هر راس متناظر با یک ناحیه قابلیت دید است و بین رئوس متناظر با دو ناحیه مجاور، یک یال جهت‌دار وجود دارد. جهت یال به طرف راس متناظر با ناحیه‌ای است که تعداد رئوس قابل دید آن کمتر است. برچسب این یال نیز همان راسی است که در یکی از این دو ناحیه دیده نمی‌شود ولی در ناحیه دیگر دیده می‌شود و در واقع راس مبدا پنجره بین دو ناحیه است.

در این گراف، به علت منطق دید در چندضلعی‌های ساده، دور وجود ندارد. همچنین رئوسی که درجه خروجی آنها صفر است متناظر با ناحیه‌هایی هستند که تعداد رئوس قابل دید آنها از همه ناحیه‌های مجاور آنها کمتر است. این ناحیه‌ها چاهک گفته می‌شوند. در شکل ۳—۴ گراف دوگان جهت‌دار متناظر با افراز مبتنی بر قابلیت دید یک چندضلعی ساده نشان داده شده است.

اگر فقط رئوس قابل دید از چاهک‌ها نگهداری شوند برای یک ناحیه غیر چاهک کافیست از راس متناظر با آن ناحیه تا رسیدن به راس متناظر با یک چاهک مسیری در گراف دوگان پیموده شود و برچسب‌های مسیر به رئوس قابل دید چاهک افزوده شوند. به این ترتیب نیازی به نگهداری رئوس قابل دید ناحیه‌های غیر چاهک نیست و می‌توان آنها را از روی گراف دوگان جهت دار و رئوس قابل دید از چاهک‌ها ساخت.

در [۶۴] و [۲۴] اثبات شده است که تعداد چاهک‌ها در افراز مبتنی بر قابلیت دید مربوط به یک چندضلعی ساده $O(n^2)$ است. در شکل ۳—۵ نیز یک چندضلعی ساده با همین تعداد چاهک نشان داده شده است.

با توجه به اینکه اندازه گراف دوگان $O(n^3)$ است و حافظه مورد نیاز برای نگهداری رئوس



شکل ۳-۵: یک چندضلعی ساده با $O(n^2)$ چاهک.

قابل دید از چاهک‌ها نیز $O(n^3)$ است پس کل حافظه مصرفی الگوریتم به $O(n^3)$ کاهش می‌یابد. زمان پیش پردازش نیز $O(n^3 \log n)$ خواهد شد. این بهبودها در هزینه پیش پردازش، تاثیری در افزایش مرتبه زمانی پرس‌وجو ندارد. برای نقطه پرس‌وجوی q با استفاده از ساختار مکان‌یابی، ناحیه در برگیرنده q در زمان $O(\log n)$ تعیین می‌شود. در گراف جهت‌دار یک مسیر از راس متناظر با این ناحیه تا رسیدن به یک چاهک دنبال می‌شود. در طول مسیر برچسب‌های یال‌ها به عنوان رئوس قابل دید نگهداری می‌شود و در محل مناسب در دنباله رئوس قابل دید از چاهک درج می‌شوند. زمان لازم برای انجام این کار نیز نسبت به اندازه $V(q)$ خطی است. جزئیات این الگوریتم در [۲۴] آمده است. نتیجه نهایی این الگوریتم بصورت قضیه زیر بیان می‌شود.

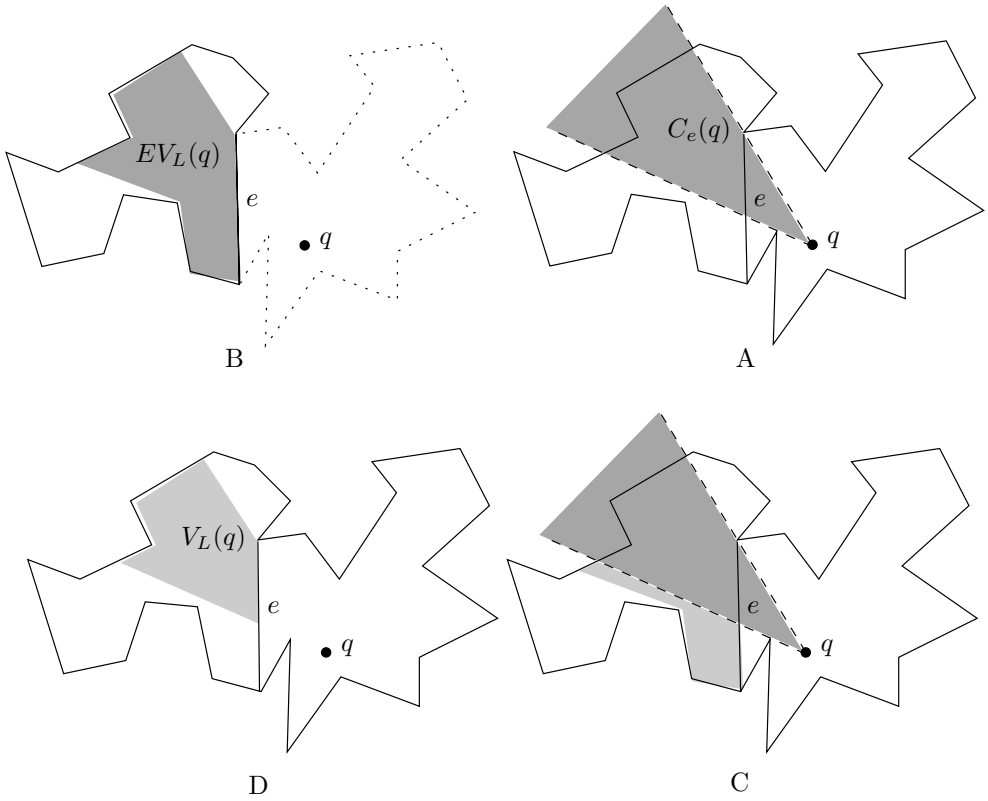
قضیه ۴ . هر چندضلعی ساده را می‌توان در زمان $O(n^3 \log n)$ به‌گونه‌ای پیش پردازش کرد که با نگهداری داده‌ساختاری با اندازه $O(n^3)$ بتوان چندضلعی قابل دید از هر نقطه پرس‌وجوی q واقع در درون آن‌را در زمان $O(|V(q)| + \log n)$ محاسبه کرد.

۳-۲ افراز بازگشتی

در این بخش الگوریتم ارائه شده در [۱۱] به اختصار معرفی می‌شود. این الگوریتم به $O(n^2 \log n)$ زمان پیش‌پردازش و $O(n^2)$ حافظه نیاز دارد و $V(q)$ را به‌ازای هر نقطه پرس‌وجوی q واقع در یک چندضلعی ساده در زمان $O(|V(q)| + \log^2 n)$ محاسبه می‌کند.

این الگوریتم بصورت بازگشتی و با تقسیم متوالی چندضلعی ساده کار می‌کند. برای این کار ابتدا روشی ارائه می‌شود که برای نقطه پرس‌وجوی q واقع در $\mathcal{P}$ که خارج از زیرچندضلعی P' از $\mathcal{P}$ قرار دارد، چندضلعی قابل دید جزئی [۳] آن که برابر با $V(q) \cap P'$ است را محاسبه می‌کند. سپس روشی برای تقسیم سلسله مراتبی $\mathcal{P}$ بیان می‌شود که مبتنی بر مثلث‌بندی متوازن است.

[۳] Partial Visibility Polygon

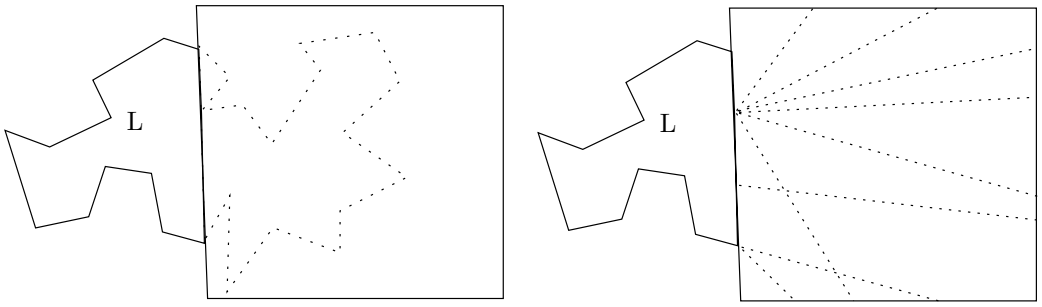


شکل ۳-۶: محاسبه چندضلعی قابل دید جزئی.

این تقسیم بگونه‌ای انجام می‌شود که به ازای هر نقطه جستجو q می‌توان $\mathcal{P}$ را به $O(\log n)$ زیرچندضلعی افراز کرد که هر یک از آنها یک گره در درخت مثلث‌بندی متوازن است و هیچ یک از این زیرچندضلعی‌ها بجز یکی از آنها که به شکل مثلث است شامل q نیستند. بدیهی است که q همه نقاط مثلث دربرگیرنده خود را می‌بینند و با محاسبه چند ضلعی‌های قابل دید جزئی برای سایر زیرچندضلعی‌ها $V(q)$ بطور کامل تعیین می‌شود.

چندضلعی قابل دید جزئی، $V_L(q)$، برای نقطه پرس‌وجوی q و چندضلعی L که درون $\mathcal{P}$ قرار دارند و $q \notin L$ بصورت $V(q) \cap L$ تعریف می‌شود. فرض کنید $\mathcal{P}$ بوسیله قطر e به دو بخش L و R تقسیم شده است. می‌خواهیم $V_L(q)$ را برای $q \in R$ محاسبه کنیم. برای این‌کار، ابتدا زاویه‌ای که q در محدوده این زاویه بخش‌هایی از L را می‌بینند تعیین می‌شود. با داشتن این زاویه $C_e(q)$ که نقاط قابل دید از q و محدود شده به این زاویه است مانند قسمت A از شکل ۳–۶ محاسبه می‌شود.

سپس، بدون در نظر گرفتن قسمت‌هایی از چندضلعی ساده که در قسمت R قرار دارند، نقاط قابل دید چندضلعی L از q و از طریق یال e مشخص می‌شود که با $EV_L(q)$ نشان داده می‌شود و در قسمت B از شکل ۳–۶ مشاهده می‌شود. همانطوریکه در قسمت‌های C و D از شکل ۳–۶ نشان



شکل ۳–۷: افراز مبتنی بر قابلیت دید نیم‌صفحه مقابل L براساس پنجره‌هایی که از رئوس L بدست می‌آیند.

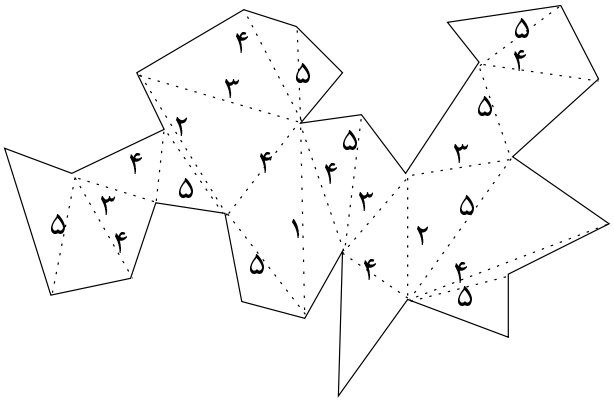
داده شده است، $V_L(q)$ برابر با $C_e(q) \cap EV_L(q)$ است و به این ترتیب مقدار آن بدست می‌آید. برای تعیین $C_e(q)$ کافی است دو ضلع زاویه α که q از طریق این زاویه یال e را می‌بینند مشخص شود. این امتدادها در الگوریتم نهایی که $V(q)$ را محاسبه می‌کند مشخص هستند و با $O(1)$ در دسترس خواهند بود.

محاسبه $EV_L(q)$ همانند آنچه در مورد افراز مبتنی بر قابلیت دید گفته شد صورت می‌گیرد. ناحیه R و در واقع نیم صفحه سمت مقابل L براساس پنجره‌هایی که بوسیله رئوس L بدست می‌آیند همانند شکل ۳–۷ به ناحیه‌های قابلیت دید افراز می‌شود. در این افراز برای هر ناحیه قابلیت دید، دنباله رئوسی از L که از نقاط آن ناحیه قابل دید هستند نگهداری می‌شود. با ایجاد یک ساختار مکان‌یابی بر روی این افراز، ناحیه دربرگیرنده q پیدا شده و دنباله رئوس قابل دید از آن ناحیه به عنوان دنباله رئوس قابل دید از q تعیین شده و با پیمایش و تصحیح این دنباله $EV_L(q)$ مشخص می‌شود.

در اینجا نیز در تحلیلی خام تعداد $O(n^2)$ پنجره خواهیم داشت که این پنجره‌ها می‌توانند در $O(n^4)$ نقطه تلاقی داشته باشند و در نتیجه $O(n^4)$ ناحیه قابلیت دید بوجود می‌آورند. ولی همانند بخش قبلی اثبات می‌شود که فقط $O(n)$ پنجره از قطر e عبور می‌کنند و در نتیجه تعداد ناحیه‌های قابلیت دید $O(n^2)$ است.

برای نگهداری دنباله رئوس قابل دید از هر ناحیه نیز همانند بخش قبلی از گراف دوگان جهت‌دار استفاده می‌شود و با یک داده‌ساختار پایدار و با حافظه $O(n^2)$ دنباله رئوس قابل دید از تمام ناحیه‌ها نگهداری می‌شود که برای هر ناحیه در زمان $O(|EV_L(q)|)$ قابل بازیابی است.

ساختار مکان‌یابی نیز با صرف زمان $O(n^2 \log n)$ و با $O(n^2)$ حافظه قابل ساخت است. با داشتن این ساختار، ناحیه‌ی دربرگیرنده هر نقطه در زمان $O(\log n)$ مشخص می‌شود. محل تقاطع $C_e(q)$ و $EV_L(q)$ نیز با جستجوی دودویی و در زمان $O(\log n)$ تعیین می‌شود. در نتیجه، $V_L(q)$ در زمان $O(|V_L(q)| + \log n)$ محاسبه می‌شود. بنابراین با صرف $O(n^2 \log n)$ زمان و $O(n^2)$ حافظه برای پیش پردازش، می‌توان چندضلعی قابل دید جزئی را در چندضلعی‌های ساده در زمان



شکل ۳–۸: مثلث بندی متوازن چندضلعی ساده.

$O(|V_L(q)| + \log n)$ محاسبه کرد.

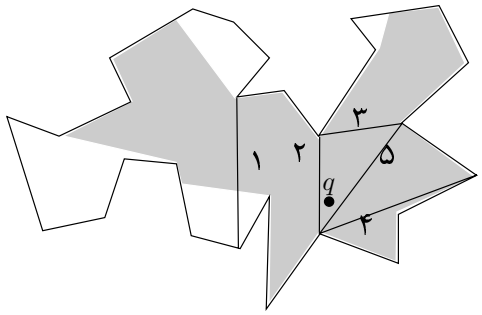
قضیه ۵ . چندضلعی ساده $\mathcal{P}$ را که بوسیله قطر e به دو بخش L و R تقسیم شده است می‌توان در زمان $O(n^2 \log n)$ و با مصرف $O(n^2)$ حافظه بگونه‌ای پیش پردازش کرد که چندضلعی قابل دید جزئی $V_L(q)$ برای هر نقطه q واقع در R را بتوان در زمان $O(|V_L(q)| + \log n)$ محاسبه کرد.

برای تعیین $V(q)$، ابتدا $\mathcal{P}$ را بصورت بازگشتی و با افزودن قطرهای داخلی به بخش‌های R و L تقسیم می‌کنیم تا در پایان به یک مثلث‌بندی برسیم. این تقسیم بازگشتی با استفاده از مثلث‌بندی متوازن که در [۲۹] بیان شده است انجام می‌گیرد. این نوع مثلث‌بندی منجر به یک درخت متوازن می‌شود که برگ‌های آن مثلث‌های تشکیل شده هستند و برای هر گره میانی، پیش‌پردازش مربوط به چندضلعی قابل دید جزئی انجام می‌شود. نمونه‌ای از این مثلث بندی متوازن در شکل ۳—۸ نشان داده شده است.

برای هر نقطه پرس‌وجو مسیر مربوط به مثلث دربرگیرنده این نقطه در درخت متوازن پیموده و در هر گره میانی از این مسیر، $V_L(q)$ نسبت به آن گره محاسبه و به هم افزوده می‌گردند. در پایان نیز مثلث در برگیرنده q به این مجموعه از چندضلعی‌ها اضافه و $V(q)$ حاصل می‌شود. این مراحل در شکل ۳—۹ نشان داده شده است.

قضیه ٦ . چندضلعی ساده P را می‌توان در زمان $O(n^2 \log n)$ و با مصرف $O(n^2)$ حافظه بگونه‌ای پیش پردازش کرد که $V(q)$ به ازای هر نقطه‌ی پرس‌وجوی q در زمان $O(|V(q)| + \log^2 n)$ قابل محاسبه باشد.

اثبات. براساس آنچه که در قضیه ۵ گفته شد و مثلث‌بندی متوازن چندضلعی، در مورد حافظه مصرفی و زمان اجرای مربوط به پیش پردازش، که به ترتیب با $S(n)$ و $T(n)$ نشان داده می‌شوند،



شکل ۳-۹: ساخت $V(q)$ با ترکیب چندضلعی‌های قابل دید جزئی در درخت مثلث‌بندی متوازن.

رابطه‌های زیر برقرارند:

$$S(n) = \max_{n/۳ \le m \le ۲n/۳}(S(m) + S(n-m)) + \Theta(n^۲),$$

$$T(n) = \max_{n/۳ \le m \le ۲n/۳}(T(m) + T(n-m)) + \Theta(n^۲ \log n).$$

بنابراین کل حافظه مصرفی $O(n^۲)$ و زمان اجرای پیش پردازش $O(n^۲ \log n)$ است. به ازای هر نقطه پرس‌وجو، مسیر مربوط به این نقطه در درخت مثلث‌بندی متوازن در زمان $O(\log n)$ مشخص می‌شود. به ازای هر گره میانی از این مسیر، $V_L(q)$ در زمان $O(\log n)$ شناسایی شده و در زمان $O(|V_L(q)|)$ تصحیح می‌شود. از آنجا که حاصل‌جمع همه $V_L(q)$ها برابر با $V(q)$ است کل زمان پرس‌وجو $O(|V(q)| + \log^۲ n)$ است. □

فصل ۴

محاسبه چندضلعی قابل دید ناظر نقطه‌ای پرس‌وجو در دامنه‌های چندضلعی گونه

در فصل ۳ مفهوم افراز مبتنی بر قابلیت دید و ناحیه‌های قابلیت دید معرفی شد. همچنین، دو الگوریتم برای تعیین چندضلعی قابل دید نقطه q واقع در درون چندضلعی ساده $\mathcal{P}$ معرفی شد که از طریق پیش‌پردازش، ناحیه‌های قابلیت دید را شناسایی و اطلاعات آن‌ها را نگهداری می‌کنند. سپس، در زمان پرس‌وجو با استفاده از این اطلاعات در زمان کمتری $V(q)$ را محاسبه می‌کنند. با این حال، هیچ یک از این روش‌ها برای چندضلعی‌های حفره‌دار قابل استفاده نیست. در این بخش الگوریتمی ارائه می‌کنیم که تعمیمی از روش‌های فوق برای بکارگیری در چندضلعی‌های حفره‌دار است.

در این روش با افزودن تعدادی قطر به چندضلعی حفره‌دار، که قطر برشی[۱] نامیده می‌شوند، و برش چندضلعی در امتداد این قطرها، یک چندضلعی ساده بدست می‌آید. ابتدا چندضلعی قابل دید از نقطه جستجو را با استفاده از یکی از الگوریتم‌های موجود بر روی این چندضلعی ساده محاسبه می‌کنیم. سپس، با انجام اصلاحاتی جهت رفع تاثیر قطرهای برشی، چندضلعی قابل دید نهایی را محاسبه می‌کنیم.

در این الگوریتم مقدار $O(n^3 \log n)$ زمان پیش‌پردازش و $O(n^3)$ حافظه مورد نیاز است. این پیش‌پردازش ما را قادر می‌سازد تا $V(q)$ را به ازای هر نقطه‌ی پرس‌وجوی دلخواه q، در زمان $O((1+h')\log n + |V(q)|)$ محاسبه کنیم که h' یک پارامتر وابسته به اندازه خروجی و نحوه‌ی پیش‌پردازش و با حداکثر مقدار h[۲] است که با بهبود الگوریتم مقدار h' به $\min(h, |V(q)|)$ کاهش می‌یابد. نتایج حاصل از این روش در [۱۳۹، ۱۴۴، ۱۴۶] منتشر شده‌اند.

---

[۱] Cut-Diagonal

بهترین الگوریتم حل این مساله با استفاده از پیش‌پردازش در [۱۰۹] ارائه شده است که با صرف زمان پیش‌پردازش $O(n \log n)$ و حافظه $O(n)$ می‌تواند $V(q)$ را برای هر نقطه پرس‌وجوی q در زمان $O(|V(q)| \log n)$ محاسبه کند. در مقایسه، الگوریتم ما دارای هزینه پیش‌پردازش بالاتر و در مقابل زمان پرس‌وجوی پایین‌تری است.

در ادامه و در بخش ۴—۱ افراز مبتنی بر قابلیت دید را برای چندضلعی‌های حفره‌دار بیان می‌کنیم. در بخش ۴—۲ جزئیات نسخه اولیه الگوریتم و تحلیل آن بیان می‌شود. سپس، الگوریتم نهایی در بخش ۴—۳ معرفی و تحلیل می‌شود.

۴-۱ افراز مبتنی بر قابلیت دید در چندضلعی‌های حفره‌دار

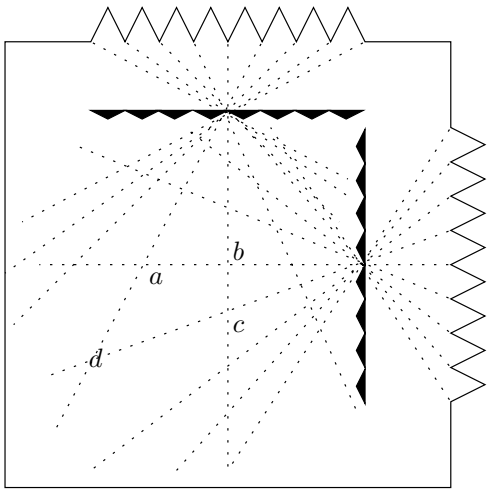
فرض کنید که $\mathcal{P}$ یک چندضلعی با حفره‌های H_1، H_2، ...، H_h است و q نقطه پرس‌وجویی است که می‌خواهیم $V(q)$ را محاسبه کنیم. افراز مبتنی بر قابلیت دید $\mathcal{P}$ همانند آن‌چه در مورد چندضلعی‌های ساده گفته شد، تعریف می‌شود. استفاده از همین روش برای چندضلعی‌های حفره‌دار نسبت به چندضلعی‌های ساده از پیچیدگی بیشتری برخوردار است.

براساس همان استدلالی که برای چندضلعی‌های ساده گفته شد، تعداد ناحیه‌های قابلیت دید در یک چندضلعی حفره‌دار $O(n^4)$ است. تعداد چاهک‌ها نیز در این نوع چندضلعی‌ها همانطور که در شکل ۴—۱ نشان داده شده است $O(n^4)$ است. در این شکل ناحیه‌ی $abcd$ دارای حداقل یک چاهک است. اندازه‌ی چندضلعی قابل دید هر چاهک نیز می‌تواند $O(n)$ باشد که این امر در مورد چندضلعی شکل ۴—۱ برقرار است. بنابراین فضای لازم برای نگهداری این اطلاعات $O(n^5)$ خواهد بود.

اگر اطلاعات مربوط به این نوع افراز برای یک چندضلعی حفره‌دار تهیه شود، با ایجاد یک ساختار مکان‌یابی مناسب بر روی این ناحیه‌ها به ازای هر نقطه‌ی پرس‌وجو ناحیه‌ی دربرگیرنده‌ی آن‌را می‌توان در زمان $O(\log n)$ پیدا کرد. اصلاح دنباله‌ی قابل دید آن ناحیه برای این‌که کاملا متناظر با چندضلعی قابل دید نقطه‌ی پرس‌وجو باشد نیز در زمان $O(|V(q)|)$ امکان‌پذیر است [۲۴]. بنابراین با مصرف $O(n^5)$ حافظه می‌توان در زمان $O(\log n + |V(q)|)$ چندضلعی قابل دید هر نقطه را محاسبه کرد.

زمان لازم برای پیش‌پردازش و ساخت داده‌ساختارهای مورد نیاز نیز $O(n^5 \log n)$ است که برابر با زمان مورد نیاز برای انجام عملیات زیر است:

ابتدا $V(r)$ به ازای هر راس r از چندضلعی $\mathcal{P}$ محاسبه می‌شود که این کار در زمان $O(n^2 \log n)$ قابل انجام است [۱۲۴]. با این کار تمام پنجره‌ها شناسایی می‌شوند. تفکیک مربوط به ناحیه‌های قابلیت دید و گراف دوگان متناظر با آن را می‌توان در زمان $O(n^4 \log n)$ ساخت [۱۹]. ساختار مکان‌یابی بر روی ناحیه‌های قابلیت دید را نیز می‌توان در زمان $O(n^4 \log n)$ ساخت [۸۵، ۹۱، ۱۱۲، ۱۱۹]. از آنجا که تعداد چاهک‌ها $O(n^4)$ است، محاسبه‌ی چندضلعی قابل



شکل ۴-۱: یک چندضلعی با $O(n^۴)$ ناحیه قابلیت دید و چاهک.

دید آنها نیازمند $O(n^۵ \log n)$ زمان است. در نتیجه کل زمان پیش پردازش $O(n^۵ \log n)$ خواهد بود. بنابراین، قضیه زیر در دست است.

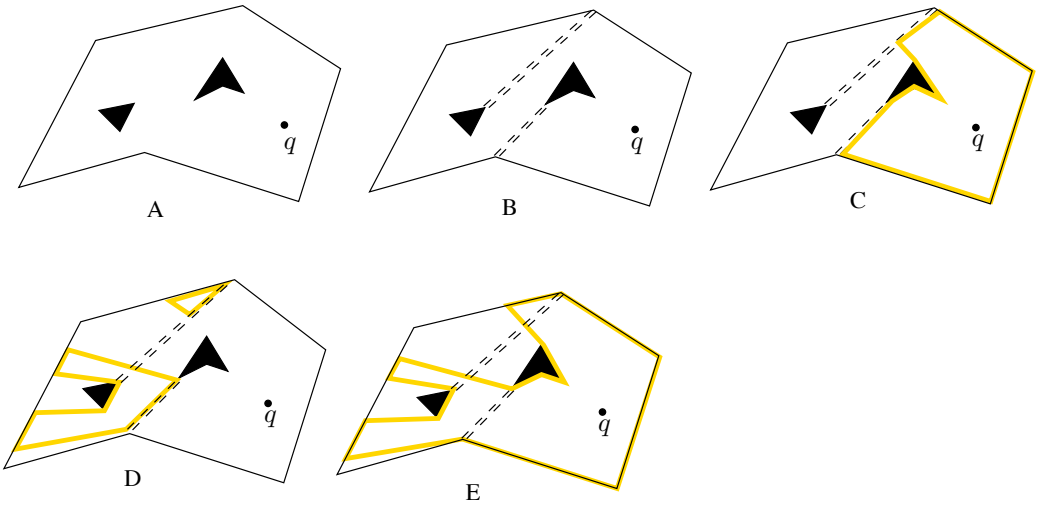
قضیه ۷ . با $O(n^۵ \log n)$ زمان پیش پردازش و $O(n^۵)$ حافظه می توان چندضلعی قابل دید هر نقطه از یک چندضلعی حفره‌دار n راسی را در زمان $O(\log n + |V(q)|)$ محاسبه کرد.

در بخش بعدی برای این مساله الگوریتمی با زمان پیش پردازش و حافظه مصرفی کمتر ارائه می‌کنیم که در مقابل، زمان پرس‌وجوی آن بیشتر از این الگوریتم است.

۴-۲ الگوریتم قطر برشی

زمان پیش پردازش و حافظه‌ی مصرفی بالای الگوریتم قبلی باعث کاهش کارایی آن در کاربردهای عملی می‌شود. در این بخش یک الگوریتم جدید ارائه می‌کنیم که دارای مرتبه‌ی زمانی و حافظه مصرفی کمتری است و در مقابل زمان پاسخ آن برای محاسبه چندضلعی قابل دید نقاط پرس‌وجو بالاتر است.

در این الگوریتم چندضلعی حفره‌دار $\mathcal{P}$ به یک چندضلعی ساده P_s تبدیل می‌شود. این کار با افزودن تعدادی قطر و برش دادن چندضلعی در امتداد این قطرهای برشی انجام می‌شود. برای محاسبه $V(q)$ ابتدا $V_s(q)$ که چندضلعی قابل دید از نقطه‌ی پرس‌وجوی q نسبت به P_s است، براساس یکی از الگوریتم‌های [۲۴] یا [۱۱] محاسبه می‌شود. سپس $V(q)$ از روی $V_s(q)$ و براساس الگوریتمی که اثر قطرهای برشی در $V_s(q)$ را از بین می‌برد بدست می‌آید.



شکل ۴-۲: مراحل محاسبه چندضلعی قابل دید.

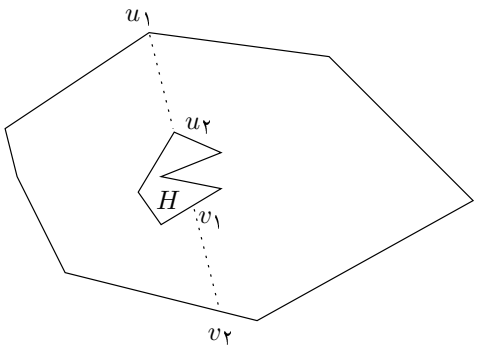
در شکل ۴—۲ نمونه‌ای از روش کار این الگوریتم نشان داده شده است. چندضلعی حفره‌دار A با اضافه شدن دو قطر برشی به چندضلعی ساده B تبدیل شده است. در واقع هر قطر برشی همانند دو ضلع از چندضلعی ساده‌ی بدست آمده عمل می‌کند. چندضلعی قابل دید نقطه q که نسبت به P_s محاسبه شده است در قسمت C نشان داده شده است. بخش‌هایی از چندضلعی اولیه که از نقطه‌ی q و از طریق قطرهای برشی قابل دید هستند مطابق قسمت D محاسبه می‌شوند. اجتماع این بخش‌ها با ناحیه‌ی بدست آمده در قسمت C کل چندضلعی قابل دید از نقطه‌ی q را تشکیل می‌دهد که در قسمت E نشان داده شده است.

جزئیات این الگوریتم در بخش‌های زیر می‌آیند.

۴-۲-۱ تبدیل چندضلعی حفره‌دار به چندضلعی ساده

برای ساخت P_s باید به ازای هر یک از حفره‌های H_i یک قطر برشی اضافه شود که یکی از رئوس H_i را به یکی از رئوس مرز بیرونی متصل می‌کند. لازم به ذکر است که این قطر برشی باید بطور کامل درون چندضلعی قرار بگیرد. برای این کار می‌توان از چپ‌ترین حفره شروع کرد که دارای چپ‌ترین راس در میان تمام حفره‌های چندضلعی است. این حفره را همواره می‌توان با یک قطر برشی از چندضلعی حذف کرد.

برای چندضلعی $\mathcal{P}$ با n راس، تعداد رئوس چندضلعی P_s برابر با $n+2h$ است و با توجه به حد بالای h که $\lfloor \frac{n-3}{3} \rfloor$ است، تعداد راس‌های P_s و در نتیجه اندازه‌ی P_s از مرتبه $O(n)$ باقی خواهد ماند.



شکل ۴-۳: جایگزینی قطر برشی u_1u_2 با v_1v_2 برای محاسبه چندضلعی قابل دید از طریق قطر u_1u_2.

تبدیل بالا را می‌توان با مثلث‌بندی چندضلعی حفره‌دار و انتخاب قطرهای برشی مناسب انجام داد که زمان اجرای آن $O(n \log n)$ است [۱۲۰].

۴-۲-۲ محاسبه دید از طریق قطرهای برشی

بخش مهمی از الگوریتم ما، محاسبه بخش‌هایی از $V(q)$ است که از طریق قطرهای برشی دیده می‌شوند. برای این‌کار از روشی که در [۱۱] ارائه شده است استفاده می‌کنیم. در آنجا نشان داده شده است که با انجام $O(n^2 \log n)$ پیش پردازش و مصرف $O(n^2)$ حافظه می‌توان برای یک چندضلعی ساده و n راسی P، که با یک قطر به دو قسمت L و R تقسیم شده است، داده ساختاری را فراهم کرد که به ازای هر نقطه‌ی جستجوی q از R، چندضلعی قابل دید جزئی آن در L، $V_L(q)$، در زمان $O(\log n + |V_L(q)|)$ محاسبه شود. این روش در بخش ۳—۲ به اختصار بیان شد.

برای استفاده از این روش، فرض کنید چندضلعی $\mathcal{P}$ مطابق شکل ۴—۳ فقط یک حفره دارد که با قطر برشی u_1u_2 به یک چندضلعی ساده تبدیل شده است. این قطر برشی را از جهت دیگر امتداد می‌دهیم تا مرز حفره و سپس مرز چندضلعی را به ترتیب در نقاط v_1 و v_2 قطع کند. چندضلعی را در امتداد پاره‌خط v_1v_2 برش می‌دهیم. در اینصورت می‌توانیم قطر برشی u_1u_2 را به عنوان یک قطر داخلی از چندضلعی ساده‌ی جدید در نظر بگیریم که در مورد آن می‌توانیم از روش [۱۱] استفاده کنیم. دلیل این امر نیز این است که برش چندضلعی در امتداد v_1v_2 هیچ تاثیری بر دید نقاط از طریق قطر u_1u_2 ندارد.

برای تعمیم این روش به بیش از یک حفره بدین ترتیب عمل می‌کنیم که هر قطر برشی را مستقل از بقیه‌ی قطرهای برشی بررسی می‌کنیم و داده‌ساختارهای متناظر با آنها را جداگانه بدست می‌آوریم، به این معنی که هنگام پردازش یک قطر برشی، بقیه‌ی قطرهای برشی به عنوان اضلاع واقعی چندضلعی در نظر گرفته می‌شوند. بنابراین زمان محاسبه‌ی چندضلعی قابل دید از یک قطر

برشی، چندضلعی قابل دید حاصل ممکن است شامل بخش‌هایی از دیگر قطرهای برشی باشد. تصحیح این امر به زمان پرس‌وجو موکول می‌شود و در آن مرحله بطور بازگشتی به ازای هر قطر برشی که در چندضلعی قابل دید ظاهر می‌شود با استفاده از داده‌ساختار مربوط به آن قطر برشی، عمل جایگزینی قطر برشی با چندضلعی قابل دید از طریق آن صورت می‌گیرد. این کار تا زمانی ادامه می‌یابد که چندضلعی قابل دید نقطه‌ی جستجو شامل هیچ قطر برشی نباشد.

۴-۲-۳ مراحل الگوریتم و تحلیل آن

الگوریتم شامل دو بخش پیش‌پردازش و پرس‌وجو است. پیش‌پردازش الگوریتم شامل موارد زیر است:

۱) افزودن قطرهای برشی و تبدیل چندضلعی اولیه $\mathcal{P}$ به چندضلعی ساده P_s بر اساس روشی که در بخش ۴—۲—۱ آمده است.

۲) پیش‌پردازش چندضلعی ساده P_s و تهیه داده ساختار مناسب بطوری که $V(q)$ نسبت به P_s بطور کارایی قابل محاسبه باشد. این کار همانند [۲۴] که در بخش ۳—۱ بیان گردید انجام می‌شود.

۳) تهیه داده‌ساختار مناسب برای محاسبه‌ی کارای دید از طریق یال‌های برشی همانند آنچه در بخش ۴—۲—۲ بیان شد.

پس از انجام پیش‌پردازش‌های بالا، در زمان پرس‌وجو $V(q)$ به ازای نقطه‌ی دلخواه q بصورت زیر محاسبه می‌شود:

ابتدا و با استفاده از داده‌ساختار تهیه شده در مرحله‌ی دوم پیش‌پردازش، چندضلعی قابل دید از q نسبت به P_s محاسبه می‌شود. با توجه به این‌که در چندضلعی بدست آمده ممکن است قطر برشی موجود باشد باید این قطرهای برشی با چندضلعی قابل دید از طریق آنها جایگزین شوند. برای این کار از داده‌ساختارهای تهیه شده در مرحله سوم پیش‌پردازش استفاده می‌شود. این مرحله از الگوریتم تا زمانی ادامه می‌یابد که هیچ قطر برشی در $V(q)$ باقی نماند. فرض کنید در یک مرحله، uu' بخشی از یک قطر برشی است که در $V(q)$ ظاهر شده است. بخش‌های قابل دید از چندضلعی ساده مربوط به این قطر برشی که از نقطه q و از طریق uu' قابل دید است را محاسبه و آنرا جایگزین uu' در $V(q)$ می‌کنیم.

بسادگی قابل اثبات است که آنچه در پایان مراحل فوق باقی می‌ماند $V(q)$ نهایی است. دلیل این ادعا این است که هر ضلع یا راس قابل دید یا بطور مستقیم از q قابل دید است یا از طریق یک یا چند قطر برشی دیده می‌شود که در هر حال پس از پایان مراحل فوق در $V(q)$ ظاهر می‌شود. نکته

دیگر پایان‌پذیر بودن این مراحل است که به دلیل ماهیت دید (دید در امتداد خط مستقیم صورت می‌گیرد)، امکان بوجود آمدن دور در جایگزینی قطرهای برشی بوجود نخواهد آمد.

تحلیل هزینه‌ی این الگوریتم از دو قضیه‌ی ۴ و ۵ که هزینه‌ی انجام بخش‌هایی از پیش‌پردازش را تعیین می‌کنند بدست می‌آید. براساس این دو قضیه لم‌های زیر در مورد هزینه پیش‌پردازش و زمان پرس‌وجوی الگوریتم ارائه شده برقرارند:

لم ۱ . زمان وحافظه مورد نیاز برای پیش‌پردازش الگوریتم ارائه شده برای یک چندضلعی n راسی با h حفره به ترتیب $O(n^3 \log n)$ و $O(n^3)$ است.

اثبات. چنانچه در بخش ۴—۲—۱ گفته شد، چندضلعی حفره‌دار را می‌توان در زمان $O(n \log n)$ با افزودن $O(h)$ قطر برشی به یک چندضلعی ساده با $O(n)$ راس تبدیل کرد. هزینه پیش‌پردازش مرحله دوم الگوریتم نیز براساس قضیه ۴ بدست می‌آید. برای هر یک از قطرهای برشی نیز باید هزینه‌ای برابر آنچه در قضیه ۵ آمده است پرداخت. با توجه به تعداد قطرهای برشی مجموع کل این هزینه‌ها نیز $O(n^3 \log n)$ زمان و $O(n^3)$ حافظه می‌شود.

بنابراین کل هزینه زمانی و حافظه‌ای پیش‌پردازش به ترتیب برابر با $O(n^3 \log n)$ و $O(n^3)$ است. . □

لم ۲ . در الگوریتم ارائه شده، $V(q)$ برای هر نقطه‌ی دلخواه q در زمان $O((1+h')\log n + |V(q)|)$ محاسبه می‌شود که h' تعداد قطرهای برشی ظاهر شده و جایگزین شده در $V(q)$ در زمان اجرای الگوریتم است.

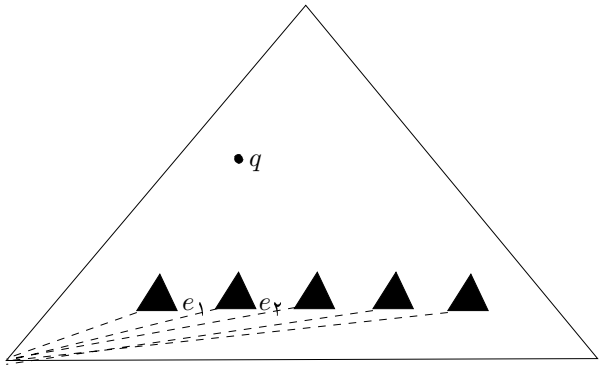
اثبات. محاسبه $V_s(q)$ با توجه به قضیه ۴ در زمان $O(\log n + |V_s(q)|)$ انجام می‌شود. به ازای هر یک از قطرهای برشی که در $V_s(q)$ ظاهر می‌شوند، مطابق با بخش ۴—۲—۲ و قضیه ۵، $O(\log n + |V_{H_i}(q)|)$ زمان لازم است تا بخشی از چندضلعی که از طریق این قطر برشی دیده می‌شود (که با $V_{H_i}(q)$ نشان داده شده است) محاسبه شود. ادغام کردن این چندضلعی‌ها با هم نیز بصورت افزایشی و با $O(1)$ انجام خواهد شد.

بنابراین، کل زمان لازم برای محاسبه $V(q)$ برابر با $O(\log n + h' \log n + h' + |V(q)|)$ است که معادل با $O((1+h')\log n + |V(q)|)$ است. □

h' یک پارامتر وابسته به خروجی و نیز موقعیت یال‌های برشی نسبت به نقطه‌ی پرس‌وجو است. بااین‌حال لم زیر در مورد کران بالای آن برقرار است.

لم ۳ . کران بالای پارامتر h' در زمان پرس‌وجوی الگوریتم برابر با h^2 است.

اثبات. می‌دانیم که قطرهای برشی بجز در نقاط انتهایی خود، با همدیگر تلاقی ندارند. بنابراین اگر قطر برشی l از نقطه‌ی q و از طریق قطر برشی l' دیده شود نقطه‌ی q دیگر نمی‌تواند l' را نیز از طریق قطر برشی l ببیند. از طرفی، نقطه‌ی q از طریق قطر برشی l' و بطور مستقیم فقط یک قطعه از



شکل ۴-۴: برای نقطه q تعداد $O(h^2)$ قطر برشی وجود دارد.

قطر l را می‌بیند. منظور از بطور مستقیم این است که هیچ قطر برشی دیگری بین l و l' قرار نگیرد که نقطه‌ی q از طریق آنها l را ببیند. بنابراین، تعداد قطعات قطرهای برشی که از نقطه‌ی q دیده می‌شوند حداکثر h^2 است. در شکل ۴—۴ برای نقطه q همین تعداد قطر برشی وجود دارد. □

براساس نتایج بالا کارایی الگوریتم ارائه شده بصورت زیر است.

قضیه ۸ . هر چندضلعی n راسی با h حفره را می‌توان با صرف $O(n^3 \log n)$ زمان و $O(n^3)$ حافظه بگونه‌ای پیش‌پردازش کرد که $V(q)$ برای هر نقطه‌ی پرس‌وجوی دلخواه q در زمان $O((1+h')\log n + |V(q)|)$ قابل محاسبه باشد. h' یک پارامتر وابسته به خروجی و پیش‌پردازش و با حداکثر مقدار $O(h^2)$ است.

مقدار h' برای یک نقطه پرس‌وجوی دلخواه q بستگی به موقعیت q و قطرهای برشی دارد و کران بالای آن فقط در شرایط خاصی رخ می‌دهد. بااین‌حال، h از مرتبه $O(n)$ است و در نتیجه h' در بدترین حالت از مرتبه $O(n^2)$ است که حداقل از نظر نظری ضعیف است. در بخش بعدی روشی برای بهبود این الگوریتم ارائه می‌کنیم که بدون افزایش هزینه پیش‌پردازش، باعث کاهش h' به $\min(h, |V(q)|)$ خواهد شد.

۴-۳ بهبود الگوریتم

همان‌طور که در شکل ۴—۴ نشان داده شده است، در برخی از موارد پردازش یک قطر برشی بطور مستقیم باعث تغییر چندضلعی قابل دید نهایی نمی‌شود. در این موارد، یک قطر برشی فقط با یک قطر برشی دیگر جایگزین می‌شود و پردازش روی این قطر برشی باید تکرار شود. به عنوان مثال، پردازش قطر برشی e_1 فقط باعث جایگزین شدن این قطر توسط قطر برشی e_2 می‌شود که اگر ما می‌توانستیم بدون پردازش e_1 مستقیما e_2 را پردازش کنیم نتیجه یکسانی بدست می‌آمد. نشان

می‌دهیم که فقط به‌علت چنین شرایطی است که ممکن است کران بالای h' رخ دهد. به‌این معنی که فقط زمانی مقدار h' از مرتبه $O(n^۲)$ است که تعداد زیادی از این پردازش‌های غیرموثر وجود داشته باشد. بنابراین اگر بتوانیم از بروز چنین شرایطی جلوگیری کنیم، کران بالای h' بطور قابل ملاحظه‌ای کاهش می‌یابد. برای این‌منظور، یک داده‌ساختار دیگر در زمان پیش‌پردازش تهیه می‌شود که به کمک آن می‌توانیم از پردازش قطرهای برشی غیرموثر جلوگیری کنیم.

قطر برشی e را موثر بر q می‌گوییم، هرگاه اختلاف $V(q)$ قبل و بعد از پردازش e بیش‌تر از جایگزینی e با یک قطر برشی دیگر باشد. درغیراین‌صورت e را غیرموثر بر q می‌گوییم. در شکل ۴—۵ قطر برشی e غیرموثر بر q است.

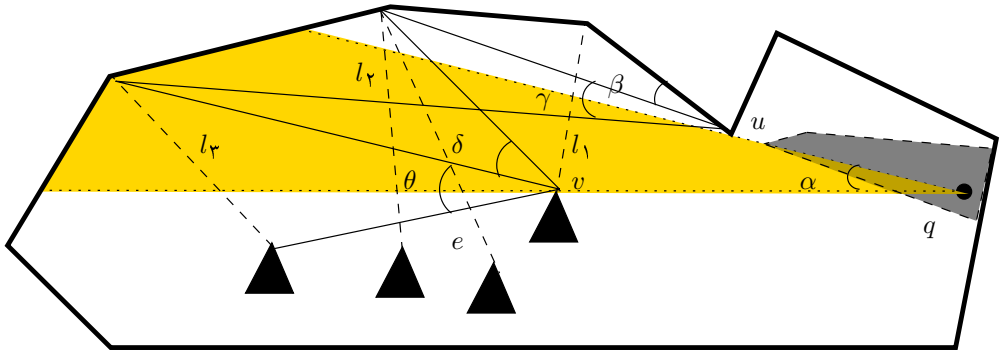
همان‌طور که در شکل ۴—۵ نشان داده شده است، نقطه پرس‌وجوی q قسمتی از یک قطر برشی $l_۱$ را با زاویه α می‌بیند که این زاویه به دو راس انعکاسی محدود شده است. این رئوس ممکن است که نقاط انتهایی $l_۱$ یا رئوسی از $\mathcal{P}$ باشند که بین q و $l_۱$ قرار دارند. در شکل ۴—۵ رئوس u و v رئوس انعکاسی مربوط به نقطه q و قطر برشی $l_۱$ هستند.

در الگوریتم ارائه شده، $l_۱$ یک پاره‌خط از $V_s(q)$ است که باید با پردازش آن، با قسمت‌هایی از چندضلعی که از q و از طریق این قطر دیده می‌شوند جایگزین شود. این پردازش باعث جایگزینی $l_۱$ با قطر برشی e می‌شود. مجددا و با پردازش e، قطر برشی $l_۲$ جایگزین e می‌شود. نهایتا پردازش $l_۲$ باعث جایگزینی این قطر برشی با قطر برشی $l_۳$ و یک یال از $\mathcal{P}$ می‌شود. بنابراین می‌توانستیم بدون پردازش $l_۱$ و e، بطور مستقیم $l_۲$ را پردازش کنیم و همان نتیجه را بدست بیاوریم. این مساله در مورد تمام نقاط q که قطر برشی $l_۱$ را از میان رئوس u و v می‌بینند و خطوط qu و qv به‌ترتیب در زاویه‌های γ و θ امتداد می‌یابند، صادق است. این ناحیه از نقاط در شکل ۴—۵ بصورت خاکستری نشان داده شده است.

برای تشخیص و نادیده گرفتن قطرهای برشی غیرموثر، به‌ازای هر زوج از رئوس انعکاسی، یک داده‌ساختار نگهداری می‌شود که به‌ازای هر نقطه پرس‌وجوی q نخستین قطر برشی موثر بر q را بطور کارایی تعیین می‌کند. برای این کار، ابتدا به‌ازای هر راس انعکاسی u، بازه‌های زاویه‌ای مختلفی که یک نقطه پرس‌وجو از طریق آن‌ها بخش‌های مختلفی از $\mathcal{P}$ را می‌بیند، تعیین می‌شوند. بازه‌های β و θ نمونه‌هایی از این بازه‌ها برای راس v در شکل ۴—۵ هستند. این بازه‌ها با اتصال v به رئوسی از $\mathcal{P}$ که از آن راس قابل دید هستند، بدست می‌آیند. این بازه‌ها یک افراز شعاعی از $\mathcal{P}$ حول v بوجود می‌آورند که به آن RD_v می‌گوییم. برای اجتناب از حالات خاص فرض شده است که هیچ ۳ راسی از $\mathcal{P}$ برروی یک خط قرار ندارند.

به ازای هر زوج از رئوس انعکاسی u و v، داده‌ساختار دیگری به نام $VR_{u,v}$ بر روی افراز شعاعی آن‌ها ساخته می‌شود. در این داده‌ساختار، $VR_{u,v}(\alpha,\beta)$ برابر با اولین قطر برشی موثر بر همه نقاطی است که امتداد دید آن‌ها در بازه‌های α از u و β از v قرار می‌گیرد. مقدار $VR_{u,v}(\alpha,\beta)$ به‌ازای همه مقادیر α از RD_u و β از RD_v تعیین و نگهداری می‌شود. این داده‌ساختارها در مرحله پیش‌پردازش تهیه می‌شوند.

در هنگام پرس‌وجو، برای نقطه q که یک قطر برشی را از میان دو راس انعکاسی u و v



شکل ۴-۵: قطر برشی e ناموثر بر q است.

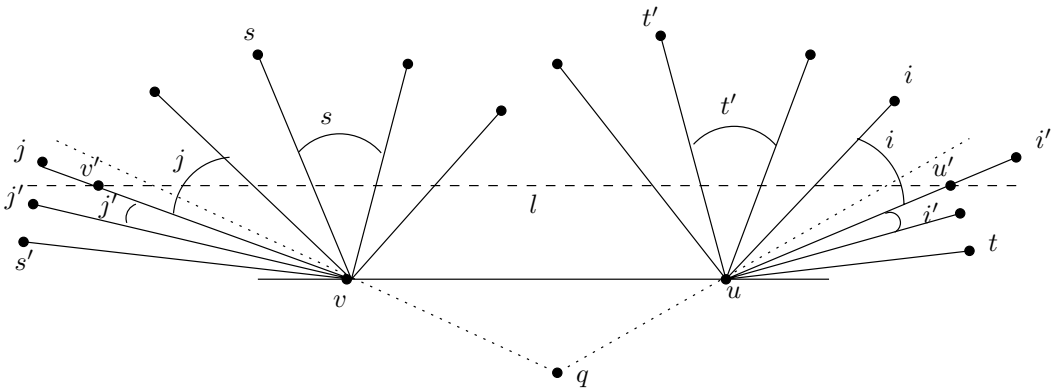
می‌بیند، بازه‌های α و β که بازه‌هایی از RD_u و RD_v هستند و به‌ترتیب امتدادهای qu و qv درون این بازه‌ها قرار می‌گیرد، تعیین می‌شود. با داشتن (α,β) مقدار $VR_{uv}(\alpha,\beta)$ به‌عنوان نخستین قطر برشی موثر بر q که باید پردازش شود، مشخص می‌گردد و به‌این‌ترتیب از پردازش قطرهای برشی غیرموثر جلوگیری می‌شود.

واضح است که $\mathcal{P}$ می‌تواند دارای $O(n)$ راس انعکاسی باشد و اندازه داده‌ساختار RD_v به‌ازای هر راس انعکاسی v می‌تواند از مرتبه $O(n)$ باشد. بنابراین تعداد $O(n^2)$ زوج راس انعکاسی u و v وجود دارد که باید VR_{uv} برای آن‌ها ساخته شود. به‌طور ابتدایی و براساس اندازه RD، اندازه هر کدام از داده‌ساختارهای VR از مرتبه $O(n^2)$ خواهد شد و در نتیجه حافظه مورد نیاز برای تمام VRها برابر با $O(n^4)$ خواهد شد.

در بخش‌های بعدی اثبات می‌کنیم که با نگهداری $O(n)$ عنصر از اعضای VR_{uv} می‌توانیم $VR_{uv}(\alpha,\beta)$ را به ازای هر مقدار دلخواه α و β محاسبه کنیم. بنابراین، حافظه مورد نیاز کل این داده‌ساختارها $O(n^3)$ خواهد شد.

۴-۳-۱ پیچیدگی حافظه‌ای VR_{uv}

فرض کنید که نقطه پرس‌وجوی q بخشی از قطر برشی l را می‌بینند که به دو راس انعکاسی u و v محدود شده است و امتدادهای qu و qv به ترتیب از درون بازه‌های زاویه‌ای α از RD_u و β از RD_v می‌گذرند. مطابق تعریف $VR_{u,v}$، مقدار $VR_{u,v}(\alpha,\beta)$ برای مقادیر متعددی از α و β برابر با خود l است که در اینصورت نیازی به نگهداری این مقادیر در $VR_{u,v}$ نیست. همچنین، می‌دانیم که امتدادهای qu و qv مطابق با تعریف بالا همدیگر را قطع نمی‌کنند و در نتیجه بازه‌های زاویه‌ای α و β همواره واگرا می‌باشند. در اینجا منظور از واگرا بودن دو بازه این است که نیم‌خط پایینیِ بازه پایینی، نیم‌خط بالاییِ بازه بالایی را قطع نمی‌کند. از این رو عناصری از $VR_{uv}(\alpha,\beta)$ که در آن‌ها α و β همگرا هستند، هیچ‌گاه مورد نیاز نخواهد بود و نیازی به نگهداری آن‌ها نیست. علاوه‌برآن،



شکل ۴-۶: فقط $O(n)$ عنصر از $VR_{u,v}$ ناتهی هستند.

مقدار برخی از عناصر $VR_{u,v}$ را می‌توان از روی مقدار سایر عناصر محاسبه کرد و در نتیجه نیازی به نگهداری این عناصر از $VR_{u,v}$ نخواهد بود (این عناصر در مورد شماره ۲ از اثبات لم بعدی بیان شده‌اند).

لم ۴ . با نگهداری $O(n)$ عنصر از اعضای $VR_{u,v}$، مقدار همه عناصر $VR_{u,v}$ را می‌توان محاسبه کرد. بنابراین، داده‌ساختار $VR_{u,v}$ را می‌توان با $O(n)$ حافظه برای هر زوج راس انعکاسی نگهداری کرد.

اثبات. رئوسی انعکاسی u و v را که $VR_{u,v}$ باید برای آن‌ها محاسبه شود را درنظر بگیرید. چنانچه پاره‌خط uv با چندضلعی $\mathcal{P}$ (علاوه‌بر نقاط انتهایی) تقاطع داشته باشد، این رئوس نمی‌توانند رئوسی باشند که یک نقطه پرس‌وجو از میان آن‌ها یک قطر برشی را می‌بینند. بنابراین نیازی به نگهداری $VR_{u,v}$ برای این موارد نیست. در ادامه فرض می‌شود که پاره‌خط uv به‌طور کامل درون $\mathcal{P}$ قرار می‌گیرد. بازه‌های RD برای رئوس u و v به‌صورت مرتب شده و درخلاف جهت عقربه‌های ساعت درنظر گرفته می‌شوند و همانند شکل ۴—۶ نامگذاری می‌شوند. براساس این نامگذاری، بازه i از بالا به پاره‌خط متصل به راس i و از پایین به پاره‌خط متصل به راس $i-1$ محدود شده است. باید توجه کرد که RD_v فقط به‌ازای رئوسی که از v قابل دید هستند، ایجاد می‌شود.

فرض کنید که مقدار $VR_{u,v}(i,j)$ غیرتهی و برابر با l باشد. این بدین معنی است که برای همه نقاط q که qu و qv در بازه‌های i و j از RD_u و RD_v امتداد می‌یابند، همه قطرهای برشی که بین l و uv قرار دارند، غیرموثر بر q هستند. همچنین، فرض کنید که i و j از هم بیشترین فاصله ممکن را دارند، به‌این‌معنی‌که بازه i' از RD_u و قبل از i وجود ندارد که مقدار $VR_{u,v}(i,j)$ غیرتهی باشد و به‌همین‌ترتیب بازه j' از RD_v و بعد از j با مقدار غیرتهی برای $VR_{u,v}(i,j')$ وجود ندارد.

این فرضیات نتایج زیر را به دنبال دارد.

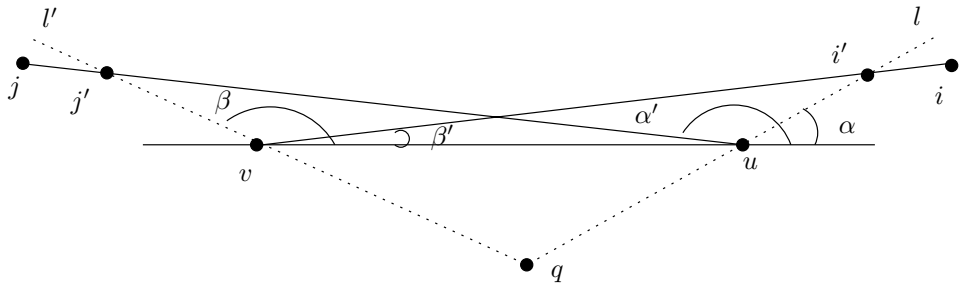
- ناحیه $uu'v'v$ شامل هیچ راسی از $\mathcal{P}$ نیست و هیچ تلاقی با یال‌های $\mathcal{P}$ نیز ندارد. این یک نتیجه بدیهی از وجود l به‌عنوان مقدار $VR_{u,v}(i,j)$ است.
- برای هر بازه s از v که قبل از j قرار دارد و هر بازه t' از u که بعد از i قرار دارد چنانچه بازه‌های s و t' غیرهمگرا باشند، $VR_{u,v}(t',s)$ برابر با l یا یک قطر برشی دیگر l' است که دورتر از l نسبت به uv قرار دارد. اگر مقدار $VR_{u,v}(t',s)$ برابر با l باشد آن‌گاه نیازی به نگهداری آن نیست و مقدار آن از روی $VR_{u,v}(i,j)$ قابل محاسبه است. برای این موارد در $VR_{u,v}$ چیزی نگهداری نمی‌شود. در زمان پرس‌وجو، هنگامی که مقدار $VR_{u,v}(\alpha,\beta)$ برای بازه‌های غیرهمگرای α و β جستجو می‌شود، مقدار $VR_{u,v}(i,j)$ به‌عنوان جواب تعیین می‌شود که i بزرگ‌ترین بازه از u قبل از α و j کوچک‌ترین بازه از v و بعد از β است که $VR_{u,v}(i,j)$ دارای مقدار است.
- به‌ازای هر بازه‌ی s از v که قبل از j قرار دارد و بازه‌های t و t' از u که به‌ترتیب قبل و بعد از بازه i قرار دارند، فقط یکی از مقادیر $VR_{u,v}(t,s)$ و $VR_{u,v}(t',s)$ می‌تواند، ناتهی باشد. دلیل این امر این است که $VR_{u,v}(t',s)$ فقط در صورتی می‌تواند ناتهی باشد که s بعد از بازه i قرار داشته باشد (مطابق شکل ۴—۶). چنانچه مقدار $VR_{u,v}(t,s)$ هم ناتهی و با مقدار l' باشد که متمایز از l است، به‌علت بیشینه بودن d قطرهای l و l' باید متقاطع باشند یا اینکه نقطه انتهایی سمت راست l که یک راس از $\mathcal{P}$ است باید بین l' و uv قرار داشته باشد. درهرصورت، هیچ یک از این موارد امکان‌پذیر نیست.
- به‌ازای هر بازه t' از u که بعد از i قرار دارد و بازه‌های s و s' از v که به‌ترتیب قبل و بعد از بازه j قرار دارند، فقط یکی از مقادیر $VR_{u,v}(t',s)$ و $VR_{u,v}(t',s')$ می‌توانند ناتهی باشند. اثبات این مطلب همانند مورد بالا است.
- به‌ازای هر زوج بازه‌های i و i' از u حداکثر یک بازه j از v وجود دارد که $VR_{u,v}(i,j)$ و $VR_{u,v}(i',j)$ ناتهی باشند. این یک نتیجه بدیهی از دو مورد بالا است.

نتیجه آخر از موارد بالا به این امر منجر می‌شود که حداکثر $2n$ عنصر از VR_{uv} ناتهی هستند و بنابراین درستی لم ثابت می‌شود. □

۴-۳-۲ ساختن داده‌ساختار $VR_{u,v}$

یک نقطه دید را که بر روی راس u قرار دارد و بازه دید آن از زاویه α تا β است درنظر بگیرید (شکل ۴—۷). برای این نقطه دید، $VR_u(\alpha,\beta)$ به‌عنوان نخستین قطر برشی موثر بر این نقطه دید تعریف می‌شود.

فرض کنید که نقطه جست‌وجوی q از میان رئوس انعکاسی u و v و بازه‌های α و β مطابق شکل ۴—۷ یک قطر برشی را می‌بینند. همچنین فرض کنید که β' بزرگ‌ترین بازه‌ای از راس v است



شکل ۴–۷: $VR_{u,v}(\alpha,\beta)$ و $VR_v(\beta,\beta')$ و $VR_u(\alpha,\alpha')$ با هم برابر هستند.

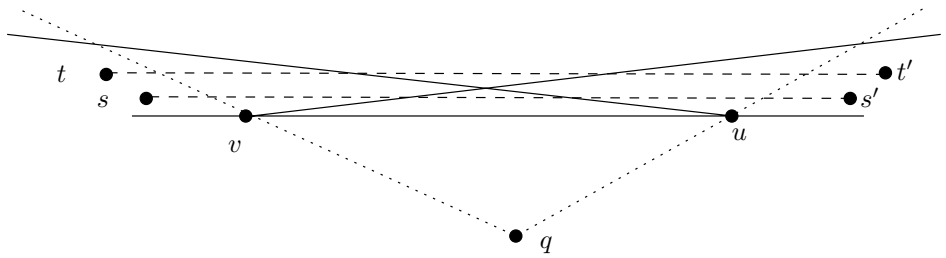
که نقطه انتهایی آن، i، قبل از خط l قرار دارد و هیچ راسی در ناحیه vui' قرار ندارد. علاوه‌بر آن، فرض کنید α' کوچک‌ترین بازه راس u است که نقطه انتهایی آن، j، بعد از خط l' قرار دارد و هیچ راسی در ناحیه vuj' قرار ندارد.

لم ۵ . اگر هیچ یک از مقادیر $VR_u(\alpha,\alpha')$ و $VR_v(\beta,\beta')$ یال فرضی uv را قطع نکنند مقدارهای $VR_{u,v}(\alpha,\beta)$، $VR_u(\alpha,\alpha')$ و $VR_v(\beta,\beta')$ با هم برابرند.

اثبات. ابتدا تساوی $VR_v(\beta,\beta')$ و $VR_u(\alpha,\alpha')$ را نشان می‌دهیم. این تساوی مبتنی بر تعریف ارائه شده برای i و j در بخش ۴–۳–۱ است. چنانچه یکی از این دو مقدار تهی باشد مقدار دیگری نیز باید تهی باشد. فرض کنید $VR_u(\alpha,\alpha')$ تهی است. این بدین معنی است که راسی مانند t وجود دارد که بطور مستقیم (نه از طریق یک قطر برشی) از q و در محدوده زاویه دید آن قابل دید است. این راس (t) درون یکی از ناحیه‌های با مرز $luj'l'$ یا $l'j'j$ قرار می‌گیرد. طبق تعریف، j کوچکترین (در مختصات قطبی حول u) راس قابل دید از u است که بعد از l' قرار دارد. بنابراین، امکان ندارد که t درون ناحیه $l'j'j$ باشد و در نتیجه t باید درون ناحیه $luj'l'$ باشد.

از طرف دیگر و طبق تعریف i، این راس (t) نمی‌تواند درون ناحیه $i'uv$ باشد. بنابراین این راس باید درون ناحیه $li'vl'$ باشد که به معنای این است که $VR_v(\beta,\beta')$ نیز تهی است. زیرا در صورتی که $VR_v(\beta,\beta')$ تهی نباشد راس t دیگر بطور مستقیم از q قابل دید نخواهد بود و از طریق قطر برشی $VR_v(\beta,\beta')$ دیده می‌شود که مغایر با فرض بالاست. بنابراین مقادیر $VR_v(\beta,\beta')$ و $VR_u(\alpha,\alpha')$ یا هر دو تهی هستند یا هیچ یک تهی نیستند.

حال فرض کنید که این دو مقدار ناتهی ولی نامساوی هستند. طبق تعریف i و j، این دو قطر برشی باید پایین زنجیره $jj'vui'i$ که در شکل ۴–۸ نشان داده شده است، قرار گیرند. از آنجا که این قطرهای برشی همدیگر را و نیز پاره‌خط uv را قطع نمی‌کنند، این قطرهای برشی باید مطابق آنچه در شکل ۴–۸ نشان داده شده است باشند. واضح است که در این شرایط ss' که قطر نزدیکتر به vu است، یا نسبت به هر دو راس u و v قطر برشی موثر است یا نسبت به هر دو ناموثر است و در هر دو صورت به معنای یکسان بودن قطرهای برشی $VR_v(\beta,\beta')$ و $VR_u(\alpha,\alpha')$ است.



شکل ۴-۸: قطرهای برشی ss' و tt' باید یکسان باشند.

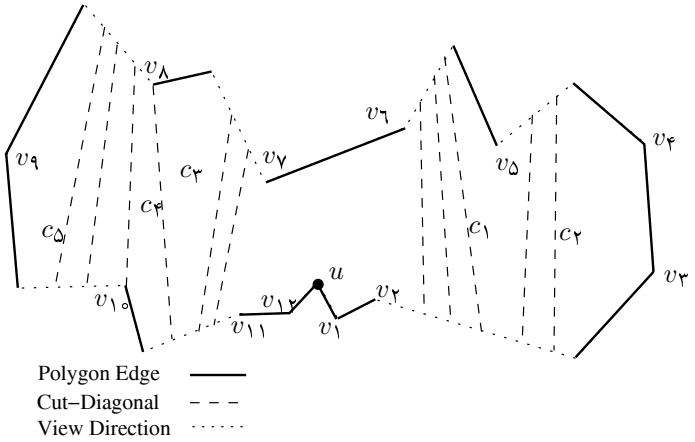
از آنجا که $VR_{u,v}(\alpha,\beta)$، در صورت وجود، باید بالای uv قرار داشته باشد، با استدلال مشابهی تساوی $VR_{u,v}(\alpha,\beta)$ و $VR_v(\beta,\beta')$ و $VR_u(\alpha,\alpha')$ را می‌توان نشان داد. □

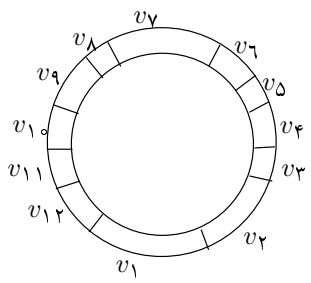
در صورتی که $VR_v(\beta,\beta')$ و $VR_u(\alpha,\alpha')$ با uv تلاقی داشته باشند یا باید همدیگر را قطع کنند یا نقطه انتهایی حداقل یکی از آنها در ناحیه vuq قرار داشته باشد. ازآنجا که دو قطر برشی همدیگر را قطع نمی‌کنند باید حالت دوم برقرار باشد که در این صورت مقدار $VR_{u,v}(\alpha,\beta)$ تهی خواهد شد. بدون کاهش کلیت مساله، فرض کنید که فقط $VR_u(\alpha,\alpha')$ با uv تلاقی دارد. در اینصورت $VR_v(\beta,\beta')$ و در نتیجه $VR_{u,v}(\alpha,\beta)$ نیز تهی خواهد بود.

با این ملاحظات، می‌توان $VR_{u,v}$ را به‌عنوان ترکیب VR_u و بازه‌های α' درنظر گرفت که نسبت به v محاسبه شده‌اند. این بازه‌های α' در یک داده‌ساختار به‌نام $VR'_{u,v}$ نگهداری می‌شوند. برای هر نقطه پرس‌وجوی q که از بین رئوس انعکاسی u و v و از طریق بازه‌های α و β می‌بینند، به‌جای استفاده از $VR_{u,v}$، ابتدا بازه α' از $VR'_{u,v}$ که متناظر با بازه β است، تعیین می‌شود. چنانچه $VR_u(\alpha,\alpha')$ تهی نباشد و قطر برشی مربوط به آن با uv تلاقی نداشته باشد این مقدار به‌عنوان نخستین قطر برشی موثر بر q تعیین می‌شود و در غیر اینصورت مقدار تهی گزارش می‌شود. می‌توان از $VR'_{v,u}$ و VR_v نیز به‌جای $VR'_{u,v}$ و VR_u استفاده کرد.

از واقعیت بالا این نتیجه بدست می‌آید که ساختن و نگهداری $VR_{u,v}$ معادل با ساختن و نگهداری زوج VR_u و $VR'_{u,v}$ یا زوج VR_v و $VR'_{v,u}$ است که نحوه ساخت و استفاده از آنها در ادامه می‌آید.

VR_u را می‌توان براساس نسخه اولیه الگوریتم که در بخش ۴—۲ ارائه شد محاسبه کرد. برای این‌کار، $V(u)$ برای هر راس u محاسبه می‌شود و در خلال این محاسبه هنگامی که یک قطر برشی در $V(u)$ ظاهر می‌شود VR_u متناظر با آن به‌هنگام می‌شود. علاوه‌برآن، RD_u می‌تواند به عنوان یک محصول جانبی از این محاسبه بدست آید. چنانچه در شکل ۴—۹ نشان داده شده است، اعضای RD_u در یک لیست مرتب حلقوی نگهداری می‌شوند. اعضای VR_u به صورت سه‌تایی مرتب (α,β,l) هستند که l قطر برشی متناظر با بازه‌های α و β است. این عناصر در دو لیست مرتب شده L_1 و L_2 نگهداری می‌شود. عناصر L_1 بر اساس ترتیب افزایشی α نگهداری می‌شوند و برای مقادیر یکسان α براساس ترتیب افزایشی β مرتب می‌شوند. عناصر L_2 بر اساس ترتیب



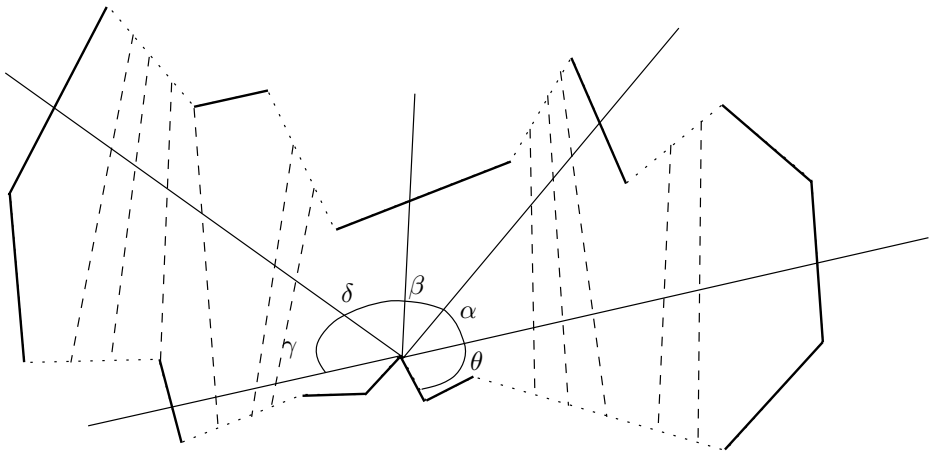


RD_u

$L_۱$	$(v_۳, v_۵, c_۲)$	$(v_۳, v_۶, c_۱)$	$(v_۸, v_{۱۱}, c_۳)$	$(v_۹, v_{۱۰}, c_۵)$	$(v_۹, v_{۱۱}, c_۴)$
$L_۲$	$(v_۳, v_۵, c_۲)$	$(v_۳, v_۶, c_۱)$	$(v_۹, v_{۱۰}, c_۵)$	$(v_۹, v_{۱۱}, c_۴)$	$(v_۸, v_{۱۱}, c_۳)$

VR_u

شکل ۴-۹: ساختن VR_u و RD_u.



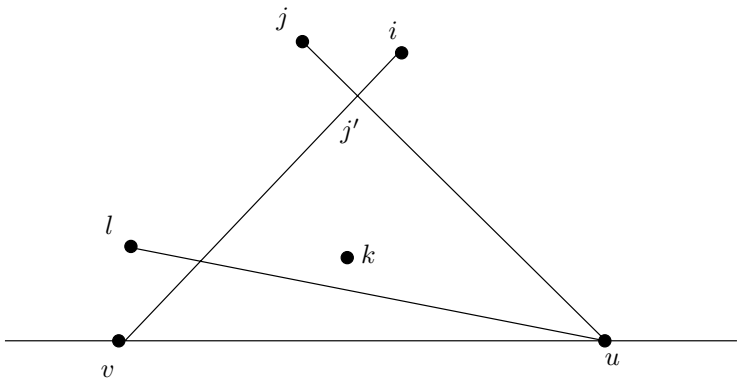
شکل ۴–۱۰: محاسبه مقادیر VR_u از روی L_1 و L_2.

افزایشی β نگهداری می‌شوند و برای مقادیر یکسان β براساس ترتیب کاهشی α مرتب می‌شوند. این لیست‌ها برای راس u در شکل ۴—۹ نشان داده شده‌اند.

با استفاده از این داده‌ساختارها، مقدار $VR_u(\alpha,\beta)$ در زمان پرس‌وجو از روی عناصر (i,j,c) و (i',j',c') از VR_u بدست می‌آید که (i,j,c) بزرگترین عنصر در L_1 است که بازه i شامل α باشد و (i',j',c') کوچک‌ترین عنصر از L_2 است که j' شامل β است. در مورد (i,j,c) و (i',j',c') چهار حالت می‌تواند وجود داشته باشد:

- β بعد از بازه j و α قبل از بازه i' قرار دارد. این حالت برای $VR_u(\alpha,\beta)$ در شکل ۴—۱۰ بوجود می‌آید. در این شرایط، مقدار $VR_u(\alpha,\beta)$ تهی است که به معنای عدم وجود قطر برشی(موثر یا غیر موثر) برای زوایای دید α و β است.
- β و α به ترتیب بعد از بازه‌های j و i' قرار دارند. این حالت برای $VR_u(\theta,\alpha)$ در شکل ۴—۱۰ بوجود می‌آید. در این شرایط، مقدار $VR_u(\alpha,\beta)$ برابر با c' است.
- β و α به ترتیب قبل از بازه‌های j و i' قرار دارند. این حالت برای $VR_u(\delta,\gamma)$ در شکل ۴—۱۰ بوجود می‌آید. در این شرایط، مقدار $VR_u(\alpha,\beta)$ برابر با c است.
- β قبل از بازه j و α بعد از بازه i' قرار دارد. این حالت برای $VR_u(\beta,\gamma)$ در شکل ۴—۱۰ بوجود می‌آید. در این شرایط، مقدار c و c' با هم برابر بوده و برابر با $VR_u(\alpha,\beta)$ است.

حال نحوه ساختن $VR'_{u,v}$ برای هر زوج راس انعکاسی u و v را بیان می‌کنیم. می‌دانیم که در $VR'_{u,v}$ به هر بازه β از RD_v یک بازه α' از RD_u منسوب می‌شود به طوری‌که α' کوچکترین بازه از u است که ضلع محدودکننده آن بعد از ضلع بازه β قرار می‌گیرد و هیچ راسی در ناحیه vuj' قرار ندارد (j' محل برخورد خط گذرنده از اضلاع بازه‌های α' و β است). به عنوان مثال، برای بازه i از



شکل ۴-۱۱: تعیین مقادیر $VR'_{u,v}$.

راس v در شکل ۴—۱۱، اگر در ناحیه vuj' هیچ راسی نباشد، بازه j از راس u منسوب می‌شود و اگر راسی مانند k در این ناحیه وجود داشته باشد بازه دیگری مانند l تخصیص داده می‌شود که شرط بالا برآورده شود.

برای ساختن $VR'_{u,v}$، بازه‌های RD_v براساس ترتیب پادساعتگرد و با شروع از بازه مربوط به خط vu بررسی می‌شوند. طبق شکل ۴—۱۱، هنگام بررسی بازه i، کوچکترین بازه j از RD_u را که شامل راس مربوط به بازه i است پیدا می‌کنیم (i و j ممکن است با یک راس یکسان تعریف شده باشند). چنانچه ناحیه vuj' شامل هیچ راسی نباشد بازه j به بازه i منسوب و در $VR'_{u,v}$ نگهداری می‌شود. درغیراینصورت، فرض کنید راس k در این ناحیه باشد که زاویه vuk کمترین مقدار خود را دارد. در این حالت، بازه l که شرط مورد نظر را دارد به بازه i منسوب می‌شود. توجه داشته باشید که راس مربوط به l ممکن است همان راس v باشد. از آنجا که بازه‌های RD_v به ترتیب پردازش می‌شوند، نیازی به جستجو برای یافتن راس k وجود ندارد و این راس در واقع راسی است که قبلا پردازش شده است. بنابراین، کافی است که در هنگام پردازش راس‌های RD_v مقدار این راس بطور مناسب بهنگام شود.

۴-۳-۳ کارایی الگوریتم بهبود یافته

کارایی نسخه بهبود یافته الگوریتم ارائه شده، در قالب لم‌ها و قضایای زیر می‌آیند.

لم ٦ . با نگهداری یک داده‌ساختار با اندازه $O(n^3)$ که در زمان $O(n^3 \log n)$ قابل ساخت است، می‌توان از پردازش قطرهای برشی غیرموثر جلوگیری کرد و نخستین قطر برشی موثر بر هر نقطه پرس‌وجو را در زمان $O(\log n)$ محاسبه کرد.

اثبات. با استدلالی مشابه آنچه برای $VR_{u,v}$ ارائه شد، اثبات می‌شود که اندازه داده‌ساختار VR_u به‌ازای هر راس انعکاسی u برابر با $O(n)$ است و به‌طور بدبینانه در زمان $O(n^2 \log n)$ و با استفاده از

الگوریتم ارائه شده در بخش ۴—۲، قابل ساخت است. همچنین RD_u که اندازه آن نیز $O(n)$ است با همین محاسبه قابل ساخت است. محاسبه $VR'_{u,v}$ به‌ازای هر زوج از رئوس انعکاسی u و v در زمان $O(n \log n)$ و با استفاده از یک خط جاروب شعاعی روی بازه‌های RD_v و با جستجوی دودویی روی بازه‌های RD_u قابل ساخت است. اندازه $VR'_{u,v}$ نیز $O(n)$ است. تعداد $O(n^2)$ زوج راس وجود دارد که بر این اساس، زمان و حافظه لازم برای ساخت و نگهداری تمام داده‌ساختارهای $VR'_{u,v}$ به‌ترتیب برابر با $O(n^3 \log n)$ و $O(n^3)$ است.

برای یافتن نخستین قطر برشی موثر برای یک نقطه پرس‌وجو، داده‌ساختارهای $VR'_{u,v}$ و VR_u باید جست‌وجو شوند که در زمان $O(\log n)$ انجام خواهد شد. بنابراین زمان پرس‌وجوی لازم برای تعیین نخستین قطر برشی موثر $O(\log n)$ است. □

لم ۷ . تعداد قطرهای برشی موثر پردازش شده با الگوریتم ارائه شده برای یک نقطه پرس‌وجوی q، $O(|V(q)|)$ است. بنابراین، کران بالای پارامتر h' نیز $O(|V(q)|)$ است.

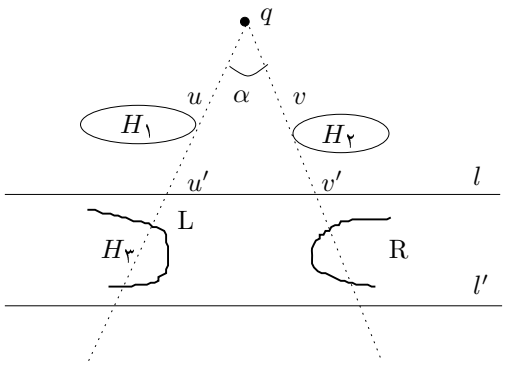
اثبات. هر قطر برشی موثر حداقل یک یال یا یک راس جدید به چندضلعی قابل دید q اضافه می‌کند که هیچ یک از این تغییرات تکراری نیست. بنابراین حد بالای تعداد قطرهای برشی موثر برابر با اندازه چندضلعی قابل دید است. □

لم بالا باعث کاهش کران بالای زمان پرس‌وجوی الگوریتم ارئه شده به $O(\log n + |V(q)| \log n + |V(q)|)$ می‌شود.

لم ۸ . تعداد قطرهای برشی موثر که با الگوریتم ارائه شده برای یک نقطه پرس‌وجوی q پردازش می‌شوند، $O(h)$ است. بنابراین، کران بالای پارامتر h' نیز $O(h)$ است.

اثبات. هر قطر برشی ممکن است از دو انتهای خود شامل قطعات موثری بر q باشد. چنانچه قطر برشی l، مطابق شکل ۴—۱۲، دارای یک بخش موثر بر q باشد باید دو حفره همانند H_1 و H_2 بین q و l قرار داشته باشد. از آنجا که $u'v'$ موثر بر q است، باید بخشهایی از $\mathcal{P}$ از q و از طریق پاره‌خط $u'v'$ دیده شود. بنابراین حداقل یکی از زنجیره‌های R و L از شکل ۴—۱۲ باید وجود داشته باشد. بدون کاهش کلیت مساله، فرض کنید L وجود دارد. حال فرض کنید بخش میانی قطر برشی دیگری مثل l' نیز موثر بر q باشد و از طریق پاره‌خط $u'v'$ توسط q دیده می‌شود. دراین‌صورت زنجیره L نمی‌تواند متعلق به مرز بیرونی $\mathcal{P}$ باشد و باید قسمتی از یک حفره دیگر مانند H_3 باشد. اگر حفره H_3 بطور کامل در زاویه α قرار نداشته باشد (همانند شکل ۴—۱۲)، دیگر حفره H_1 نمی‌تواند از سمت راست نقشی که برای پاره‌خط $v'u'$ داشت در مورد قطر برشی دیگری داشته باشد. اگر H_3 بطور کامل درون زاویه α قرار بگیرد، موثر بودن قطعه پاره‌خطهای میانی l' را می‌توان به حساب H_3 گذاشت.

بنابراین هر حفره ممکن است باعث موثر شدن حداکثر دو قطعه میانی از قطرهای برشی شود. در نتیجه تعداد قطعات قطرهای برشی موثر بر q از مرتبه $O(h)$ است.



شکل ۴-۱۲: هر حفره حداکثر باعث موثر شدن دو قطعه میانی از قطرهای برشی می‌شود.

□

بنابراین با این بهبود در الگوریتم و پیش‌پردازش انجام شده، کارایی الگوریتم به‌صورت زیر است.

قضیه ۹ . هر چندضلعی حفره‌دار $\mathcal{P}$ با n راس و h حفره را می‌توان با صرف $O(n^۳ \log n)$ زمان و $O(n^۳)$ حافظه به گونه‌ای پیش‌پردازش کرد که چندضلعی قابل دید از هر نقطه پرس‌وجوی q در زمان $O((۱ + h') \log n + |V(q)|)$ قابل محاسبه باشد. $|V(q)|$ برابر با اندازه چندضلعی قابل دید و h' یک پارامتر وابسته به پیش‌پردازش و اندازه خروجی با حداکثر مقدار $\min(h, |V(q)|)$ است.

۴-۴ نتیجه‌گیری

در این فصل یک الگوریتم جدید برای تعیین $V(q)$ در یک چندضلعی حفره‌دار ارائه شد که برای نقاط پرس‌وجوی متعدد q بتواند $V(q)$ را بصورت کارایی محاسبه کند. برای این منظور، چندضلعی مرجع را در زمان $O(n^۳ \log n)$ پیش‌پردازش می‌کنیم و داده‌ساختارهایی با اندازه $O(n^۳)$ بدست می‌آوریم که به کمک آنها بتوانیم $V(q)$ را در زمان $O((۱ + h') \log n + |V(q)|)$ محاسبه کنیم که $h < \min(h, |V(q)|)$.

در این الگوریتم از روش افراز مبتنی بر قابلیت دید برای پیش‌پردازش استفاده شده است. و با تبدیل چندضلعی حفره‌دار به یک چندضلعی ساده، مساله در محیط جدید حل و سپس تصحیحات لازم روی آن انجام می‌شود.

در فصل بعدی از این روش برای محاسبه و نگهداری چندضلعی قابل دید یک ناظر متحرک استفاده شده است. نکته مهمی که در ارتباط با این الگوریتم وجود دارد وجود پارامتر h' است که تلاش برای حذف آن می‌تواند نقطه شروعی برای بهبود این الگوریتم باشد.

فصل ۵

نگهداری چندضلعی قابل دید ناظر نقطه‌ای متحرک

در فصل‌های قبلی نحوه محاسبه $V(q)$ برای یک ناظر نقطه‌ای در محیط‌های دو بعدی بیان شد. در این فصل روش نگهداری و بهنگام‌سازی $V(q)$ را زمانی که q حرکت می‌کند بیان می‌کنیم. این مساله را از دو جهت بررسی می‌کنیم: نخست فرض می‌کنیم که فقط نگهداری دنباله رئوس و اضلاع قابل دید مد نظر است. در این حالت کافیست که ساختار ترکیبیاتی $V(q)$ را بهنگام داشته باشیم و هر وقت که در اثر حرکت، این ساختار نیاز به تغییر داشته باشد آنرا بهنگام‌سازی کنیم. بنابراین، در این روش مقدار دقیق قابل دید از هر ضلع مد نظر نیست.

برای حل این مساله الگوریتم‌های متعددی ارائه شده است که معمولا شامل دو مرحله پیش‌پردازش و زمان اجرا هستند. در مرحله پیش‌پردازش اطلاعات لازم را تهیه و نگهداری می‌کنند تا در هنگام حرکت، در زمان بهینه تغییرات را شناسایی و در $V(q)$ اعمال کنند. معیارهای اصلی مقایسه این الگوریتم‌ها شامل زمان و حافظه پیش‌پردازش، زمان پردازش رخدادها و تعداد رخدادهایی است که پردازش می‌کنند. هیچ‌یک از الگوریتم‌های قبلی هم‌زمان از هر دو جهتِ زمان پردازش و تعداد رخدادها بهینه نیستند. در این فصل الگوریتم ارائه شده در فصل ۴ را بکار می‌گیریم و براساس آن روشی برای حل این مساله ارائه می‌کنیم که با هزینه پیش‌پردازش بالاتر، هم زمان پردازش رخداد آن بهینه ($O(\log n)$) است و هم تعداد رخدادهایی که پردازش می‌کند کمینه است (فقط رخدادهای موثر بر $V(q)$ را پردازش می‌کند) [۱۴۷].

سپس، حالت دیگری که در هر لحظه شکل دقیق $V(q)$ مد نظر است بررسی می‌شود. در این مساله ما می‌خواهیم که شکل دقیق $V(q)$ و نه ساختار ترکیبیاتی آن را، زمانی که q حرکت می‌کند در هر لحظه دلخواه داشته باشیم. به عنوان مثال، می‌خواهیم $V(q)$ را در خلال حرکت q در فاصله‌های زمانی نزدیک به هم بر روی یک صفحه نمایش نشان دهیم. در این مساله نگهداری ساختار ترکیبیاتی $V(q)$ نیازمند یک پردازش مجدد برای تعیین اندازه بخش قابل دید از هر ضلع است و

درهر مرحله باید $V(q)$ مجددا بطور کامل رسم شود. برای اجتناب از این مساله الگوریتمی ارائه می‌شود که مقدار دقیق $V(q)$ را در طول حرکت نگهداری می‌کند و در هر زمان که رسم $V(q)$ اتفاق می‌افتد اصلاحات لازم، در زمان بهینه خطی نسبت به تعداد اصلاحات، در آن اعمال می‌شود [۱۳۸].

۵-۱ مقدمه

در این فصل، مساله نگهداری چندضلعی قابل دید از یک ناظر نقطه‌ای متحرک q واقع در یک دامنه چندضلعی گونه بررسی و مطالعه می‌شود. در این محیط، q در هر جهتی و با سرعتی که بوسیله یک تابع جبری با درجه ثابت بیان می‌شود حرکت می‌کند. این مساله کاربردهای زیادی در حوزه‌های گرافیک کامپیوتری، بازی‌های کامپیوتری، دید ماشین، روباتیک و برنامه‌ریزی حرکت دارد. به همین جهت، این مساله توسط محققان زیادی بررسی و مطالعه شده و الگوریتم‌های متعددی برای آن ارائه شده است.

الگوریتم‌های اصلی این مساله در [۱۱، ۴۹، ۷۶، ۷۷، ۱۰۰، ۱۱۵، ۱۱۶] ارائه شده‌اند. این الگوریتم‌ها برمبنای مفاهیم مختلفی شامل افراز مبتنی بر قابلیت دید، مجتمع قابلیت دید، گراف مماس قابلیت دید، داده‌ساختارهای جنبشی[۱]، نقشه توپولوژیکی[۲] و تفکیک شعاعی[۳] ارائه شده‌اند.

در همه این روش‌ها، یک مرحله پیش‌پردازش وجود دارد که طی آن اطلاعات دیداری محیط تهیه می‌شود. این اطلاعات برای افزایش کارایی محاسبه مقدار اولیه $V(q)$ و نیز بهنگام‌سازی آن در هنگام حرکت استفاده می‌شوند. همچنین، صفی از رخدادهایی که در هنگام حرکت اتفاق می‌افتند و باعث تغییر $V(q)$ می‌شوند نیز تهیه می‌شود که برای شناسایی و انجام تغییرات بکار می‌رود. معیارهای اصلی ارزیابی کارایی و مقایسه این الگوریتم‌ها زمان و حافظه مصرفی مرحله پیش‌پردازش، تعداد رخدادهای پردازش شده و زمان پردازش هر رخداد است.

الگوریتم‌های ارائه شده در [۱۱۵] و [۱۱۶] مبتنی بر مجتمع قابلیت دید هستند که در زمان $O(n\log n+k)$ قابل ساخت است (k اندازه گراف مماس قابلیت دید است که بین $\Omega(n)$ و $O(n^2)$ متغیر است). با داشتن مجتمع قابلیت دید یک صحنه دو بعدی، $V(q)$ برای هر q دلخواه در زمان $O(|V(q)|\log n)$ قابل ساخت است [۱۰۹]. براساس الگوریتم ارائه شده در [۱۱۵]، با داشتن مجتمع قابلیت دید و مقدار اولیه $V(q)$، هر رخداد تغییر دید هنگام حرکت q در زمان $O(\log^2(|V(q)|))$ قابل پردازش است. همچنین، اگر مسیر حرکت q از قبل مشخص باشد، با صرف $O(|V(q)|\log(|V(q)|))$ زمان پیش‌پردازش اضافی می‌توان هر رخداد را در زمان $O(\log(|V(q)|))$ پردازش کرد. الگوریتم ارائه شده در [۱۱۶] برای ناظری که در امتداد یک پاره‌خط pq حرکت می‌کند استفاده می‌شود. در این شرایط و پس از محاسبه مقدار اولیه $V(q)$ طبق الگوریتم [۱۱۵]، تغییرات $V(q)$ در زمان

[۱] Kinetic Data Structure

[۲] Topological Map

[۳] Radial Subdivision

$O(\max(|V(q)|, |V(p.q)|))$ قابل انجام هستند که $|V(p,q)|$ برابر با تعداد رخدادهای تغییر دید در خلال حرکت از p به q در امتداد pq است. مهم‌ترین مشکل این الگوریتم‌ها این است که رخدادهای پردازش شده لزوما باعث تغییر $V(q)$ نمی‌شوند و به همین دلیل پردازش زائد محسوب می‌شوند.

الگوریتم ارائه شده در [۴۹] مبتنی بر گراف قابلیت دید است. مرحله پیش‌پردازش این الگوریتم شامل ساختن گراف قابلیت دید صحنه است که در زمان $O(m \log n)$ قابل انجام است که m اندازه گراف قابلیت دید و مقداری بین $O(n)$ و $O(n^2)$ است. سپس، هر رخداد تغییر دید در زمان $O(\log m)$ پردازش می‌شود. مشکل اصلی این روش این است که تعداد رخدادهای ناموثر بر $V(q)$ در این روش بسیار بالاست. این نقیصه درروش ارایه شده در [۱۰۰] تا حدودی مرتفع شده است. در روش اخیر، فقط بخشی از گراف قابلیت دید که مرتبط با q است ساخته می‌شود که به آن تفکیک شعاعی یا نقشه توپولوژیکی گفته می‌شود. اندازه تفکیک شعاعی برابر با $O(n)$ است و در زمان $O(n \log n)$ ساخته می‌شود. با داشتن این ساختار، هر رخداد تغییر دید در زمان $O(\log^2 n)$ پردازش می‌شود. با وجود پایین بودن هزینه پیش‌پردازش در این روش، هنوز هم بخشی از رخدادهای پردازش شده غیرضروری هستند و علاوه‌برآن زمان پردازش رخدادها نیز بهینه نیست. این الگوریتم در [۷۶] بهبود یافت بگونه‌ای که با همان هزینه پیش‌پردازش، زمان متوسط پردازش هر رخداد تغییر دید $O(\log n)$ است. این روش فقط برای موانع محدب ارائه شد که در [۷۷] به موانع مقعر نیز تعمیم یافت.

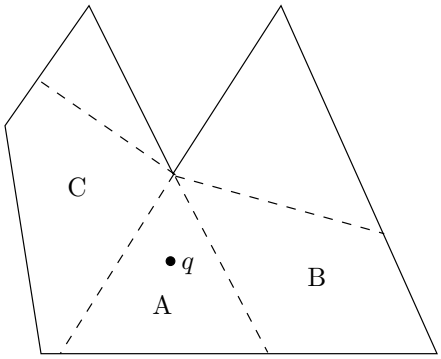
الگوریتم ارائه شده در [۱۱] فقط برای چندضلعی‌های ساده قابل استفاده است و براساس افراز مبتنی بر قابلیت دید و درخت کوتاه‌ترین مسیر است. مرحله پیش‌پردازش این الگوریتم در زمان $O(n \log n)$ و با حافظه $O(n)$ انجام می‌شود. سپس، هر رخداد در زمان $O(\log^2(|V(q)|))$ پردازش می‌شود.

چنانچه در بالا بیان شد، در هیچ‌یک از الگوریتم‌های بالا بطور هم‌زمان تعداد و زمان پردازش رخدادها بهینه نیست. همچنین، محاسبه مقدار اولیه $V(q)$ در این الگوریتم‌ها، بخصوص برای حالتی که q بصورت تصادفی و بطور مکرر از جاهای مختلف انتخاب می‌شود، بهینه نیست.

در بخش ۵—۲ یک الگوریتم برای این مساله ارائه می‌شود که فقط رخدادهای ضروری را پردازش می‌کند و زمان پردازش آن نیز بهینه است. همچنین، با توجه به این‌که این الگوریتم مبتنی بر الگوریتم ارائه شده در فصل ۴ است، محاسبه مقدار اولیه $V(q)$ نیز بصورت کارا انجام می‌شود.

تا اینجا، منظور از $V(q)$ و بهنگام‌سازی آن ساختار ترکیبیاتی بود که شامل دنباله یال‌ها و راس‌های قابل دید است. با داشتن این دنباله و با یک پیمایش خطی روی آن می‌توان مرز دقیق چندضلعی قابل دید را محاسبه کرد. اما در برخی کاربردها، تعیین و نمایش مرز دقیق چندضلعی قابل دید الزامی است. همچنین در هر جابجایی کوچک ناظر لازم است که مرز جدید محاسبه و اصلاحات لازم در آن اعمال شود، هر چند که این جابجایی ساختار ترکیبیاتی $V(q)$ را تغییر ندهد. این مساله را محاسبه و نگهداری چندضلعی قابل دید دقیق می‌نامیم که در این فصل معرفی و برای آن الگوریتمی کارا ارائه می‌شود.

در ادامه، در بخش ۵—۲ الگوریتم نگهداری ساختار ترکیبیاتی $V(q)$ ارائه می‌شود. در



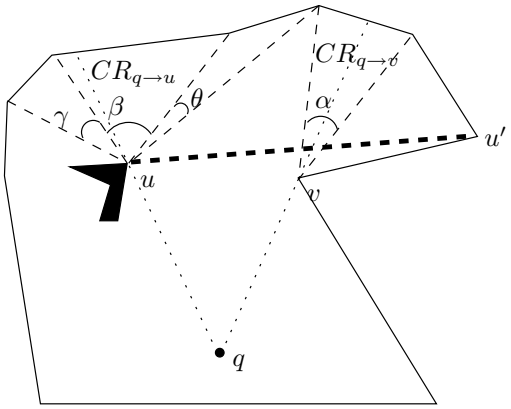
شکل ۵-۱: تا زمانی که q درون ناحیه A قرار دارد $V(q)$ ثابت است، و به محض این‌که q وارد B یا C می‌شود ساختار ترکیبیاتی $V(q)$ تغییر می‌کند.

بخش ۵—۳ مساله چندضلعی قابل دید دقیق معرفی می‌شود. سپس، مرحله پیش‌پردازش الگوریتم حل این مساله در بخش ۵—۴ توضیح داده و در بخش ۵—۵ الگوریتم مربوط ارائه می‌شود. نحوه پردازش رخداد این الگوریتم و تحلیل نظری آن در بخش ۵—۶ بیان می‌شود. در پایان، نتایج پیاده‌سازی این الگوریتم در بخش ۵—۷ ارائه می‌شود.

۵-۲ نگهداری ساختار ترکیبیاتی چندضلعی قابل دید ناظر نقطه‌ای

مقدار اولیه $V(q)$ براساس الگوریتم ارائه شده در فصل ۴ محاسبه می‌شود. زمانی که q حرکت می‌کند، $V(q)$ بطور پیوسته تغییر می‌کند ولی در این تغییرات ساختار ترکیبیاتی $V(q)$ ثابت می‌ماند. با گذشت زمان، رئوس و یال‌های جدیدی در دید q قرار می‌گیرند و برخی از رئوس و یال‌هایی که قبلا دیده می‌شدند، دیگر قابل دید نیستند. از آنجا که هدف ما نگهداری ساختار ترکیبیاتی $V(q)$ است، کافی است که فقط زمان‌هایی که این تغییرات پیش می‌آیند شناسایی شده و تغییرات لازم در $V(q)$ اعمال شود و نیازی به اصلاح $V(q)$ بصورت پیوسته نیست.

طبق تعریف ناحیه قابلیت دید در افراز مبتنی بر قابلیت دید، ساختار ترکیبیاتی همه نقاط یک ناحیه یکسان است. بنابراین، مادام که q درون یک ناحیه حرکت می‌کند نیازی به تغییر آن نیست. در مقابل، هر وقت که ناظر از یک ناحیه وارد ناحیه مجاور آن می‌شود ساختار ترکیبیاتی $V(q)$ تغییر می‌کند. این مطلب در شکل ۵—۱ نشان داده شده است. مادام که q در ناحیه A قرار دارد، $V(q)$ ثابت است ولی وقتی که q وارد یکی از ناحیه‌های B یا C شود یکی از راس‌هایی که قبلا قابل دید بود، دیگر دیده نمی‌شود و باید از $V(q)$ حذف شود. بسته به جهت و سرعت حرکت ناظر، زمانی که این رخداد اتفاق می‌افتد قابل محاسبه و پیش‌بینی است. این نوع رخداد را رخداد افرازی می‌نامیم و ناحیه‌ای که q درون آن قرار دارد با CR_q نشان می‌دهیم.



شکل ۵-۲: رخداد قطر برشی زمانی اتفاق می‌افتد که $CR_{q\to v}$ یا $CR_{q\to u}$ تغییر کند.

چنانچه در فصل ۴ گفته شد، افراز مبتنی بر قابلیت دید برای چندضلعی ساده حاصل از برش دادن چندضلعی حفره‌دار بوسیله قطرهای برشی تهیه می‌شود. بنابراین، رخدادهای افرازی فقط رخدادهای تغییر دید را نسبت به این چندضلعی ساده نشان می‌دهند. بااین‌حال، رخدادهای تغییر دید دیگری از طریق تغییر دید از میان قطرهای برشی اتفاق می‌افتد. این نوع رخدادها را رخداد قطر برشی می‌نامیم.

یک رخداد برشی زمانی اتفاق می‌افتد که دنباله راس‌ها و یال‌های قابل دید از طریق یک قطر برشی تغییر کند. در شکل ۵—۲، چندضلعی اولیه با افزودن قطر برشی uu' به یک چندضلعی ساده تبدیل شده است. ناظر q بخش‌هایی از چندضلعی را از طریق این قطر برشی می‌بیند. رئوس u و v زاویه دید q را از طریق این قطر برشی محدود می‌کنند و α و β بازه‌هایی از RD_v و RD_u هستند که امتدادهای qv و qu درون آنها قرار می‌گیرند. این بازه‌ها به ترتیب با $CR_{q\to v}$ و $CR_{q\to u}$ نشان داده می‌شوند. وقتی که q حرکت می‌کند، مادام که این بازه‌ها ثابت هستند دنباله راس‌ها ویال‌های قابل دید از طریق این قطر برشی تغییر نمی‌کند.

بنابراین، یک رخداد قطر برشی زمانی اتفاق می‌افتد که یکی از امتدادهای qu و qv از بازه خود خارج و وارد بازه مجاور شود. به بیانی دیگر، یک رخداد قطر برشی زمانی پیش می‌آید که $CR_{q\to v}$ یا $CR_{q\to u}$ تغییر کند. به عنوان مثال، زمانی که امتداد qu در شکل ۵—۲ از بازه β خارج و وارد بازه θ یا γ می‌شود یک رخداد قطر برشی اتفاق می‌افتد. با داشتن جهت و سرعت حرکت q می‌توان زمان وقوع این نوع از رخدادها را پیش‌بینی کرد.

قضیه ۱۰ . رخدادهای قطر برشی و افرازی شامل همه رخدادهای تغییر ساختار ترکیبیاتی چندضلعی قابل دید ناظر نقطه‌ای در یک چند حفره‌دار هستند.

اثبات. واضح است که در یک چندضلعی ساده فقط رخدادهای افرازی اتفاق می‌افتد و طبق تعریف افراز مبتنی بر قابلیت دید، هر تغییر دید در $V(q)$ متناظر با یک رخداد افرازی است.

از طرف دیگر، طبق تعریف بازه‌های دید مربوط به رئوس انعکاسی، دید ناظر q فقط زمانی تغییر می‌کند که $CR_{q\to v}$ یا $CR_{q\to u}$ تغییر کند که u و v رئوس محدود کننده دید q از طریق قطر برشی مورد نظر هستند و هر یک از این تغییرات متناظر با یک رخداد قطر برشی است.

درنتیجه، کلیه تغییرات دید یک ناظر نقطه‌ای متحرک در یک چندضلعی حفره‌دار با رخدادهای افرازی و رخدادهای قطر برشی هم‌ارز هستند. □

۵-۲-۱ ساخت صف اولیه رخدادها

مقدار اولیه $V(q)$ براساس الگوریتم ارائه شده در فصل ۴ محاسبه می‌شود. برای تشخیص تغییرات دید، صفی از رخدادهای آینده تهیه می‌شود. این رخدادها شامل یک رخداد افرازی و $2h'$ رخداد قطر برشی است که زمان وقوع آنها بستگی به جهت و سرعت حرکت ناظر دارد. فرض کنید که ناظر در امتداد یک خط مشخص و با سرعت مشخص حرکت می‌کند. قبل از شروع حرکت و برای موقعیت اولیه q صف رخدادها بصورت زیر ساخته می‌شود.

زمان محاسبه مقدار اولیه $V(q)$، CR_q نیز بدست می‌آید. با فرض بالا، یال e از CR_q که توسط q قطع می‌شود با یک جستجوی دودویی تعیین می‌شود. زمان بروز این تقاطع محاسبه می‌شود و به عنوان تنها رخداد افرازی با اولویت زمان رخداد آن در صف قرار می‌گیرد.

رخدادهای قطربرشی مربوط به قطرهای برشی هستند که از q قابل دید هستند. در زمان محاسبه مقدار اولیه $V(q)$، هر جا که یک قطر برشی در $V(q)$ ظاهر می‌شود هنگام محاسبه دید از طریق این قطر برشی $CR_{q\to u}$ و $CR_{q\to v}$ نیز بدست می‌آیند که با دانستن جهت و سرعت حرکت q زمان بروز رخدادهای مربوط به آنها قابل محاسبه است. زمان‌های وقوع این رخدادها محاسبه می‌شود و به عنوان اولویت آنها در صف قرار می‌گیرند.

بدیهی است که زمان وقوع این رخدادها بستگی به جهت حرکت ناظر دارد و در صورتی‌که جهت حرکت ناظر تغییر کند، کلیه این رخدادها مجددا باید محاسبه شوند.

۵-۲-۲ پردازش رخدادها

وقوع یک رخداد افرازی به معنی خروج q از CR_q قبلی و ورود به یک ناحیه دیگر است. بدیهی است که CR_q جدید مجاور با CR_q قبل است و یال مشترک آنها در زمان محاسبه و پیش‌بینی رخداد مشخص شده است. برای پردازش این رخداد، رخداد قبلی از صف وقایع حذف و رخداد جدید که مربوط به CR_q جدید است به صف اضافه می‌شود.

بهنگام‌سازی $V(q)$ مهم‌ترین کاری است که هنگام پردازش هر رخداد باید انجام شود. تغییر CR_q به این معنی است که یک راس و یالی که قبلا از q قابل دید نبودند اکنون قابل دید هستند، یا اینکه راس و یالی که قبلا قابل دید بودند دیگر توسط q دیده نمی‌شوند. جهت و برچسب یال

مربوط به گراف دوگان افراز مبتنی بر قابلیت دید مشخص کننده نوع تغییر $V(q)$ در این رخداد است. وقتی $V(q)$ بزرگ‌تر شده باشد راس و یال جدید در محل مناسب $V(q)$ درج می‌شوند، و وقتی که دید کم شده باشد راس و یال مربوط از $V(q)$ حذف می‌شوند.

ممکن است یالی که تازه اضافه یا حذف شده است یک قطر برشی باشد. در چنین شرایطی اگر یک قطر برشی باید به $V(q)$ اضافه شود با استفاده از الگوریتم پایه‌ای، بخش‌هایی از چندضلعی که از طریق این قطر برشی دیده می‌شوند محاسبه شده و به $V(q)$ اضافه می‌شوند. همچنین، رخداد قطر برشی مربوطه نیز محاسبه و به صف رخدادها اضافه می‌شود. لازم به یادآوری است که محاسبه بخش‌های قابل دید از یک قطر برشی ممکن است منجر به ظهور قطرهای برشی جدیدی شود که این کار باید بصورت بازگشتی ادامه پیدا کند تا زمانی که هیچ قطر برشی در $V(q)$ باقی نماند.

یک رخداد برشی نیز زمانی اتفاق می‌افتد که $VR_{q\to u}$ مربوط به راس انعکاسی u که محدود کننده دید q از طریق یک قطر برشی است تغییر کند. در این حالت $CR_{q\to u}$ به یکی از دو بازه مجاورش منتقل می‌شود. برای پردازش این رخداد، آنرا از صف رخدادها حذف می‌کنیم و رخداد جدیدی که متناظر با $CR_{q\to u}$ جدید است به صف اضافه می‌شود. علاوه‌برآن، $V(q)$ دقیقا مشابه آنچه در مورد رخداد افرازی گفته شد، بهنگام می‌شود.

مساله دیگری که باید در زمان پردازش رخدادها به آن توجه شود زمانی است که ناظر از روی یک قطر برشی عبور می‌کند. در این حالت صف مربوط به رخدادها، دقیقا همانند آنچه برای موقعیت اولیه q انجام می‌شود، مجددا محاسبه می‌شود.

۵-۲-۳ تحلیل الگوریتم

در این بخش، کارایی این الگوریتم را از نظر اندازه صف رخدادها، زمان پردازش یک رخداد، ساختن اولیه صف رخدادها و بهنگام‌سازی این صف و نیز تاثیر تغییر حرکت ناظر تحلیل می‌کنیم.

لم ۹ . طول صف رخدادها در یک زمان مشخص برابر است با $۲h' + ۱$، که h' تعداد قطرهای برشی موثر بر دید ناظر در آن لحظه است.

اثبات. همواره یک رخداد افرازی برای ناظر وجود دارد که مربوط به موقعیت جاری ناظر در چندضلعی ساده شده است. علاوه‌برآن، به ازای هر یک از قطرهای برشی موثر در دید دو رخداد برشی نگهداری می‌شود که تعداد کل آنها برابر است با $۲h'$. □

لم ۱۰ . در صورت مشخص بودن CR_q، رخداد افرازی متناظر با آن در زمان $O(\log(|V(q)|))$ قابل محاسبه است.

اثبات. برای محاسبه رخداد افرازی متناظر با CR_q، باید یالی از CR_q که در ادامه حرکت q قطع می‌شود و زمان این برخورد محاسبه شود. این یال با جستجوی دودویی روی یال‌های ناحیه CR_q براساس جهت حرکت q بدست می‌آید. زمان این برخورد نیز با توجه به مشخص بودن و محدود بودن درجه تابع حرکت q در زمان ثابت محاسبه می‌شود. بنابراین برای اثبات این لم کافی است نشان دهیم که تعداد یالهای CR_q برابر است با $O(|V(q)|)$.

هر یک از یال‌های CR_q متناظر با دو راس از $\mathcal{P}$ هستند که حداقل راس نزدیکتر به q از q قابل دید است. هر یک از این راس‌ها نیز حداکثر مربوط به دو یال از مرز ناحیه CR_q است. دلیل این امر این است که امتدادهای همه پاره‌خط‌های یک راس که در تجزیه قابلیت دید ظاهر می‌شوند همگی از آن راس عبور می‌کنند و به همین دلیل هر ناحیه قابلیت دید بین دو خط از این مجموعه قرار می‌گیرد، و در نتیجه به ازای یک راس حداکثر دو پاره‌خط متصل به آن در مرز یک ناحیه قابلیت دید ظاهر می‌شود. □

لم ۱۱ . درصورت مشخص بودن $CR_{q\to u}$، رخداد قطر برشی متناظر با آن در زمان ثابت محاسبه می‌شود.

اثبات. در مورد این رخدادها کافی است که زمان منتقل شدن امتداد qv به یکی از بازه‌های مجاور $CR_{q\to v}$ مشخص شود. با فرضیاتی که در مورد جهت و سرعت حرکت ناظر داریم این محاسبه در $O(۱)$ انجام می‌شود. □

قضیه ۱۱ . صف رخدادهای مربوط به موقعیت اولیه q در زمان $O(h' \log(h') + \log(|V(q)|))$ قابل محاسبه و ساخت است.

اثبات. زمانی که ما با استفاده ازالگوریتم پایه‌ای، $V(q)$ را محاسبه می‌کنیم، CR_q و همه $CR_{q\to u}$ها را نیز در خلال همان فرایند و بدون صرف هزینه بیشتر بدست می‌آوریم. با استفاده از این واقعیت و نتایج لم‌های ۱۰ و ۱۱ قضیه اثبات می‌شود. □

وقتی جهت حرکت ناظر تغییر می‌کند، صف رخدادها باید از اول ساخته شود که با توجه به اینکه در آن لحظه CR_q و همه $CR_{q\to v}$ مشخص هستند، براساس نتیجه قضیه ۱۱ این صف را می‌توان در زمان $O(h' \log(h') + \log(|V(q)|))$ ساخت. نتیجه زیر را داریم.

لم ۱۲ . وقتی جهت حرکت ناظر تغییر می‌کند، صف جدید رخدادها در زمان $O(h' \log(h') + \log(|V(q)|))$ قابل ساخت است.

در زمان وقوع یک رخداد، علاوه‌بر بهنگام‌سازی صف رخدادها که هزینه آنها در لم‌های ۱۰ و ۱۱ بیان شد، $V(q)$ هم باید نسبت به این رخداد بهنگام شود. قضیه زیر کران بالای پردازش هر رخداد را مشخص می‌کند.

قضیه ۱۲ . هر رخداد افرازی یا قطر برشی در زمان $O(\log n)$ پردازش می‌شود.

اثبات. از آنجا که مقدار CR_q یا $CR_{q\to v}$ برای رخداد مربوط مشخص است، رخداد جدید در زمان $O(\log(|V(q)|))$ برای رخداد افرازی و در زمان $O(1)$ برای رخداد قطر برشی محاسبه می‌شود. همچنین، حذف رخداد قبلی و افزودن رخداد جدید با توجه به اندازه صف رخدادها در زمان $O(\log h')$ انجام می‌شود.

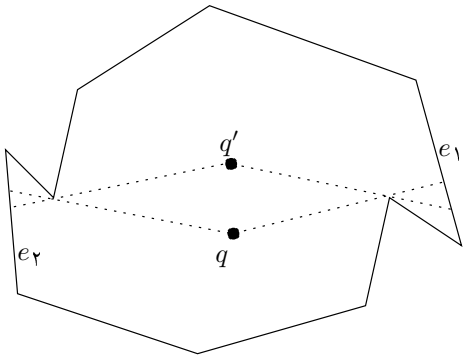
پردازش رخداد با حذف یک راس و یال از $V(q)$ یا افزودن یک راس و یال به آن انجام می‌شود. با توجه به موجود بودن ترتیب $V(q)$، هر دوی این عملیات در زمان $O(\log(|V(q)|))$ قابل انجام هستند. چنانچه یال حذف شده یک قطر برشی باشد، رخدادهای متناظر با آن که حداکثر دو رخداد قطر برشی هستند در زمان $O(\log(h'))$ قابل حذف هستند. از طرفی، اگر یال اضافه شده یک قطر برشی باشد باید مشخص شود که q از طریق این قطر برشی چه بخش‌هایی را می‌بیند که به $V(q)$ اضافه شوند. از آنجا که در ابتدای ظاهر شدن این قطر برشی فقط یک راس و یال از طریق آن قابل دید است، این محاسبه در زمان ثابت انجام می‌شود. همچنین تعیین بازه‌های اولیه $CR_{q\to v}$ برای دو رخداد متناظر با این قطر برشی در زمان $O(\log n)$ قابل انجام هستند و در نتیجه کل این پردازش در زمان $O(\log n)$ انجام‌پذیر است. قابل ذکر است که در این‌جا نیازی به فراخوانی بازگشتی برای محاسبه فضای قابل دید از یک قطر برشی، به دلیل غیر موثر بودن قطرهای میانی، وجود ندارد. در نتیجه، کل زمان لازم برای پردازش هر رخداد $O(\log n)$ است. □

نکته مهمی که در ارتباط با این روش وجود دارد این است که در این الگوریتم فقط رخدادهایی پردازش می‌شوند که بر $V(q)$ موثر باشند و علاوه‌برآن، زمان پردازش هر رخداد نیز مناسب است. البته این کارایی بالای زمان پرس‌وجو به علت هزینه پیش‌پردازش بالایی است که در ابتدای الگوریتم پرداخت می‌شود.

۵-۳ چندضلعی قابل دید دقیق

مهم‌ترین مساله‌ای که در ارتباط با الگوریتم‌های معرفی شده قبلی برای نگهداری ساختار ترکیبیاتی $V(q)$ مطرح است این است که در این روش‌ها مقدار دقیق چندضلعی قابل دید بدست نمی‌آید. بنابراین اگر بخواهیم مقدار دقیق قابل دید از هر یال و در نتیجه شکل دقیق $V(q)$ را محاسبه کنیم لازم است که یک پیمایش خطی روی $V(q)$ صورت بگیرد تا این مقادیر دقیق محاسبه شوند. این کار باید در هر مرتبه تکرار شود که باعث کاهش کارایی استفاده از این الگوریتم‌ها در کاربردهای واقعی می‌شود.

علاوه‌برآن، پیچیدگی بالای پیاده‌سازی این الگوریتم‌ها مانع استفاده واقعی از آنها می‌شود. به همین دلیل، معمولا در عمل چندین الگوریتم ساده استفاده می‌شود که در هر بار نیاز برای محاسبه و رسم $V(q)$، آنرا از اول و بدون توجه و استفاده از شکل قبلی آن محاسبه می‌کنند. برای افزایش



شکل ۵-۳: هنگام حرکت ناظر از q به q'، چندضلعی قابل دید دقیق فقط در نزدیکی یال‌های e_1 و e_2 تغییر می‌کند.

کارایی این روش‌ها نیز بجای استفاده از الگوریتم‌های بهینه از تسهیلات سخت‌افزاری برای افزایش سرعت استفاده می‌شود.

در این بخش روشی برای نگهداری شکل دقیق $V(q)$ ارائه می‌دهیم که بجای ساختار ترکیبیاتی $V(q)$، شکل دقیق آن‌را نگهداری می‌کند و در نتیجه هنگام درخواست (مثلا برای نمایش $V(q)$ روی صفحه) نیاز به پردازش اضافه‌ای نخواهد بود. این روش در واقع ترکیبی از روش‌های کاربردی و نظری برای حل این مساله است که ضمن ملاحظات کاربردی از دید نظری نیز بهینه است.

برای دستیابی به این هدف، یک لیست، $C(q)$، نگهداری می‌شود که متناظر با یال‌هایی از $V(q)$ است که بطور کامل از q قابل دید نیستند. این لیست، بخش‌هایی از $V(q)$ را نشان می‌دهد که در اثر حرکت q تغییر می‌کنند. به عنوان مثال، برای ناظر q نشان داده شده در شکل ۵—۳، $C(q)$ برابر با یالهای e_1 و e_2 است. زمانی که ناظر به نقطه q' می‌رسد، چندضلعی دقیق $V(q)$ فقط با محاسبه مجدد بخش‌های قابل دید یال‌های e_1 و e_2 محاسبه می‌شود. این گونه بهنگام‌سازی‌ها که باعث تغییر در ساختار ترکیبیاتی $V(q)$ نیز نمی‌شوند کاری است که اغلب باید در کاربردهای واقعی انجام شود. بنابراین، با داشتن مقدار دقیق $V(q)$، برنامه‌نویس کاربردهای واقعی نیازی به محاسبه مجدد آن نخواهد داشت و بسادگی می‌تواند در هر لحظه از آن استفاده کند.

با حرکت بیشتر q، برخی از یال‌ها ممکن است از $C(q)$ حذف شوند یا یال‌های جدیدی به آن اضافه شوند. برای انجام کارای این تغییرات، از یک نسخه توسعه‌یافته گراف قابلیت دید استفاده می‌کنیم. این داده‌ساختار در مرحله پیش‌پردازش تهیه می‌شود و شامل همه اطلاعات لازم برای پردازش و انجام هر یک از این تغییرات در زمان $O(1)$ است. اندازه این داده‌ساختار با اندازه گراف قابلیت دید صحنه مرجع برابر است که مقدار آن بین $\Omega(n)$ و $O(n^2)$ است. با استفاده از این روش، تغییرات مربوط به $V(q)$، هنگام حرکت q، در زمان خطی نسبت به تعداد تغییرات $V(q)$ محاسبه و اعمال می‌شود. چنانچه فاصله زمانی بین موقعیت قبلی q و موقعیت فعلی آن کوچک باشد این

تغییرات از مرتبه $O(|C(q)|)$ است و به همین جهت زمان بهنگام‌سازی مقدار دقیق $V(q)$ کاملا بهینه است.

در مقایسه با روش‌های موجود، در بهترین الگوریتم موجود $V(q)$ بصورت دنباله‌ای از یال‌های قابل دید از q نگهداری می‌شود و بهنگام‌سازی $V(q)$ در زمان $O(\log n)$ انجام می‌شود. اما در این روش‌ها محاسبه مقدار دقیق $V(q)$، که در کاربردهای واقعی مورد نیاز است، نیازمند یک پیمایش دیگر روی $V(q)$ است که در زمان $O(|V(q)|)$ انجام می‌شود. ازآنجاکه $C(q)$ یک زیردنباله از $V(q)$ است، کارایی روش جدید از این الگوریتم بهتر است. به عنوان مثال، فرض کنید q در امتداد st حرکت می‌کند و در طول حرکت در k لحظه زمانی می‌خواهیم $V(q)$ دقیق را بر روی صفحه نمایش نشان دهیم و ساختار ترکیبیاتی $V(q)$ در این مسیر l بار تغییر می‌کند. پیچیدگی زمانی بهترین الگوریتم موجود برای کل این مسیر $O(k\overline{|V(q)|}+l\log n)$ است که $\overline{|V(q)|}$ متوسط اندازه $V(q)$ در طول این مسیر است. در مقابل، در الگوریتم جدید بعد از تهیه مقدار اولیه $V(q)$، کل زمان پردازش در طول این مسیر برابر با $O(k\overline{|C(q)|}+l)$ است که $\overline{|C(q)|}$ برابر با متوسط طول $C(q)$ در این مسیر است. مشخص است که این زمان پردازش نسبت به بهترین الگوریتم موجود بهتر است و علاوه‌برآن، کاملا وابسته به اندازه خروجی است و به اندازه صحنه مرجع یعنی n بستگی ندارد.

علاوه‌برآن، درصورت تغییر جهت حرکت، در الگوریتم‌های قبلی $O(n\log n)$ زمان نیاز است که صف رخدادها بهنگام شود، درحالی‌که در این الگوریتم نیازی به پردازش اضافه نیست.

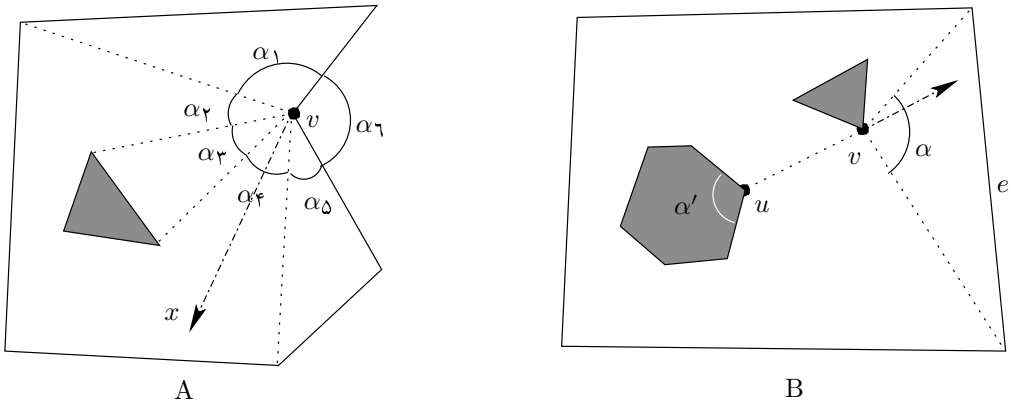
این الگوریتم به سادگی قابل پیاده‌سازی است و نتایج پیاده‌سازی انجام شده نیز موید کارایی و سهولت استفاده از آن در کاربردهای واقعی که تعداد موانع زیادی در صحنه وجود دارد است.

در ادامه، ابتدا مرحله پیش‌پردازش الگوریتم و سپس خود الگوریتم، تحلیل و نتایج پیاده‌سازی آن بیان می‌شود.

۵-۴ پیش‌پردازش برای نگهداری چندضلعی قابل دید دقیق

در مرحله پیش‌پردازش الگوریتم، مجموعه‌ای از اطلاعات دیداری صحنه مرجع تهیه می‌شود که باعث تسهیل در اجرای مرحله اصلی الگوریتم می‌شود. در این الگوریتم، مقدار اولیه $V(q)$ براساس هر یک از الگوریتم‌های موجود [۱۳، ۷۱، ۱۰۹، ۱۲۴، ۱۳۹] قابل محاسبه است. در اینجا روش ارائه شده در [۱۳] را استفاده می‌کنیم که بدون نیاز به پیش‌پردازش، $V(q)$ را در زمان $O(n\log n)$ محاسبه می‌کند.

در این مرحله داده‌ساختارهای لازم برای پیش‌بینی و تشخیص و نیز پردازش رخدادهای تغییر دید را تهیه می‌کنیم. در واقع ما می‌خواهیم همه اطلاعات مورد نیاز برای پردازش رخدادهای دیداری در زمان ثابت را داشته باشیم. این اطلاعات را می‌توانیم در قالب نسخه بهبود یافته‌ای از گراف قابلیت دید صحنه جمع‌آوری و نگهداری کنیم. به همین منظور، فرض می‌کنیم که یال‌های مجاور هر راس از گراف قابلیت دید بصورت پادساعت‌گرد حول آن راس مرتب شده‌اند. این یالهای



شکل ۵–۴: نسخه بهبود یافته گراف قابلیت دید.

شعاعی فضای اطراف هر راس را به تعداد بازه زاویه‌ای افراز می‌کنند. با داشتن این افزار، در زمان ثابت می‌توانیم از یک بازه به بازه مجاور آن حرکت کنیم.

این لیست مرتب شده از شعاع‌ها را برای راس v با R_v و بازه‌ای که شعاع $\overrightarrow{vx}$ درون آن قرار دارد با $R_v(\overrightarrow{vx})$ نشان می‌دهیم. به عنوان مثال، R_v و $R_v(\overrightarrow{vx})$ مربوط به راس v در قسمت A از شکل ۵–۴ به ترتیب برابر با $\langle \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6 \rangle$ و α_4 است.

مورد دیگری که در این داده‌ساختار باید باشد این است که در زمان ثابت بتوانیم مشخص کنیم که امتدادهای هر یال uv از گراف قابلیت دید از دو طرف درون کدام بازه از R_u و R_v قرار می‌گیرند. این بازه‌ها را با $R_v(\overrightarrow{uv})$ و $R_u(\overrightarrow{vu})$ نشان می‌دهیم. به عنوان مثال، $R_v(\overrightarrow{uv})$ و $R_u(\overrightarrow{vu})$ در قسمت B از شکل ۵–۴ به ترتیب برابر با α و α' هستند. با استفاده از لیست‌های مرتب R_u و R_v، این بازه‌ها در زمان لگاریتمی قابل محاسبه هستند. ولی در هنگام پردازش رخدادها ما نیاز داریم که این کار را در زمان ثابت انجام دهیم. به همین دلیل، $R_v(\overrightarrow{uv})$ و $R_u(\overrightarrow{vu})$ مربوط به هر یال uv از گراف قابلیت دید را باید در مرحله پیش‌پردازش محاسبه و در این داده‌ساختار نگهداری کنیم.

یک مساله دیگر که برای پردازش سریع رخدادها نیاز داریم این است که در زمان ثابت، بتوانیم اولین یالی از صحنه را که توسط امتدادهای هر یال گراف قابلیت دید قطع می‌شود مشخص کنیم. این یال‌ها را برای یال uv از گراف قابلیت دید با $e_{\overrightarrow{uv}}$ و $e_{\overrightarrow{vu}}$ نشان می‌دهیم. در قسمت B از شکل ۵–۴، $e_{\overrightarrow{uv}}$ برابر است با e و ازآنجا که $\overrightarrow{vu}$ بعد از راس u از داخل صحنه خارج می‌شود، $e_{\overrightarrow{vu}}$ برابر با تهی در نظر گرفته می‌شود. برای اینکه در زمان حرکت و هنگام پردازش رخدادها بتوانیم این مقادیر را برای یک یال uv در زمان ثابت محاسبه کنیم، مقادیر $e_{\overrightarrow{uv}}$ و $e_{\overrightarrow{vu}}$ برای هر یال uv از گراف قابلیت دید در مرحله پیش‌پردازش محاسبه و آنها را به عنوان بخشی از گراف قابلیت دید نگهداری می‌کنیم.

نهایتا، برای سهولت پیمایش و حرکت بین این داده‌ساختارها، پیوندهای مستقیمی بین یال‌ها و رئوس گراف و یال‌ها و رئوس متناظر با آنها در صحنه و نیز بازه‌های مجاور طبق تعریف بالا ایجاد

می‌کنیم.

این نسخه از گراف قابلیت دید را گراف قابلیت دید تقویت شده می‌نامیم. طبق توضیحات بالا، در گراف قابلیت دید تقویت شده مقدار ثابتی اطلاعات اضافه‌تر نسبت به گراف قابلیت دید در هر یال نگهداری می‌کنیم. بنابراین، نتیجه زیر را داریم.

لم ۱۳ . پیچیدگی حافظه‌ای گراف قابلیت دید و گراف قابلیت دید تقویت شده با هم برابر هستند. در نتیجه، حافظه مورد نیاز در مرحله پیش‌پردازش این الگوریتم برای یک صحنه n راسی بین $\Omega(n)$ و $O(n^2)$ است.

برای ساختن گراف قابلیت دید تقویت شده، با استفاده از یکی از الگوریتم‌های موجود گراف قابلیت دید صحنه را می‌سازیم و اطلاعات اضافه را به آن اضافه می‌کنیم. برای این کار می‌توانیم از الگوریتم ساده‌ی ارائه شده در [۸۹] استفاده کنیم. دراینصورت، پس از محاسبه گراف قابلیت دید، لیست یال‌های مجاور همه راس‌ها را در زمان $O(|e|\log n)$ مرتب می‌کنیم. سپس، $R_v(\overrightarrow{uv})$ و $e_{\overrightarrow{uv}}$ را به ازای همه یال‌های گراف قابلیت دید در زمان $O(e+n)$ محاسبه می‌کنیم که دلیل آن در ادامه همین بخش خواهد آمد. بنابراین، کل زمان پیش‌پردازش برابر است با $O(n^2 \log n)$ و بسادگی قابل پیاده‌سازی و استفاده در کاربردهای واقعی است.

روش فوق از نظر زمانی بهینه نیست. بجای آن می‌توانیم از الگوریتمی که در [۸۳] ارائه شده است برای ساخت گراف قابلیت دید تقویت شده استفاده کنیم. در این الگوریتم، یال‌های مجاور هر راس بصورت مرتب شده حول آن راس محاسبه و نگهداری می‌شوند (لم ۲.۴ از [۸۳]). با داشتن این لیست‌های مرتب شده R_v برای همه رئوس v، براساس روش ارائه شده در زیر، با یک پیمایش خطی روی این لیست‌ها می‌توانیم $R_v(\overrightarrow{vu})$ را به ازای همه یال‌های vu محاسبه و نگهداری کنیم.

فرض کنید $E_v = \langle vu_1, vu_2, \ldots, vu_k\rangle$ لیست یال‌های مجاور راس v در گراف قابلیت دید محاسبه شده توسط الگوریتم [۸۳] است. ترتیب یال‌ها در E_v متناظر با ترتیب پادساعت‌گرد این یال‌ها حول v است. لیست $R_v = \langle \alpha_1, \alpha_2, \ldots, \alpha_k\rangle$ را بگونه‌ای می‌سازیم که α_i $(1 \le i < k)$ زاویه تعریف شده در جهت پادساعت‌گرد بین یال‌های vu_i و vu_{i+1} است و α_k این زاویه برای یال‌های vu_k و vu_1 است. با داشتن دو لیست E_v و R_v، $R_v(\overrightarrow{u_iv})$ را به ازای $1 \le i < k$ بدین صورت محاسبه می‌کنیم: ابتدا $\alpha_i = R_v(\overrightarrow{u_1v})$ را با استفاده از یک پیمایش خطی بدیهی یا جستجوی دودویی روی R_v پیدا می‌کنیم. حال فرض کنید که یک لیست مجازی دیگری بصورت $R'_v = \langle \alpha_i, \alpha_{i-1}, \ldots, \alpha_1, \alpha_k, \alpha_{k-1}, \ldots, \alpha_{i+1}\rangle$ وجود دارد. دراین‌صورت، با پیمایش همزمان دو لیست R'_v و E_v مقدار $R_v(\overrightarrow{u_iv})$ به ازای همه یال‌های vu_i از E_v محاسبه می‌شوند.

اطلاعات مربوط به $e_{\overrightarrow{uv}}$ نیز با انجام تغییر کوچکی در الگوریتم [۸۳] بدست می‌آیند. این تغییر به این ترتیب انجام می‌شود که هر پاره‌خط e_i که از راس v و از میان دو یال مجاور vu_i و vu_{i+1} قابل دید باشد در زمان محاسبه گراف قابلیت دید نگهداری می‌شود. با توجه به نحوه محاسبه گراف قابلیت دید در این الگوریتم، این تغییر بدون انجام پردازش اضافی در دسترس است. در اینجا از بیان جزئیات بیشتر این روش به دلیل نیاز به توضیح کامل الگوریتم [۸۳] صرف‌نظر شده است.

این یال‌های e_i در واقع همان مقادیر مربوط به $e_{\overrightarrow{uv}}$ هستند که به معنی اولین یالی از صحنه است که توسط امتداد $\overrightarrow{uv}$ قطع می‌شود.

در این روش، هر یال گراف قابلیت دید $O(1)$ بار پردازش می‌شود و بنابراین نتیجه زیر را داریم.

لم ۱۴ . گراف قابلیت دید تقویت‌شده در زمان پیش‌پردازش $O(e+n\log n)$ قابل ساخت است.

نتایج بدست آمده در این بخش را در قالب قضیه زیر جمع‌بندی می‌کنیم.

قضیه ۱۳ . اندازه گراف قابلیت دید تقویت شده یک صحنه با n راس برابر است با $O(e+n)$ و در زمان $O(e+n\log n)$ قابل ساخت است که e تعداد یال‌ها در گراف قابلیت دید این صحنه است و مقدار آن حداکثر $O(n^2)$ است.

بنابراین زمان و حافظه مورد نیاز در مرحله پیش‌پردازش به ترتیب برابر با $O(e+n\log n)$ و $O(e+n)$ است.

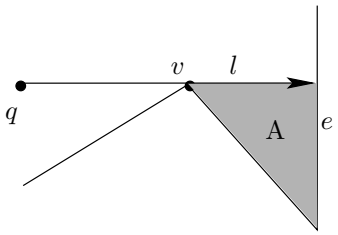
۵-۵ الگوریتم نگهداری چندضلعی قابل دید دقیق

در روش‌های نظری موجود، ساختار ترکیبیاتی $V(q)$ نگهداری و در هنگام حرکت بهنگام می‌شود. این ساختار فقط در زمان‌های خاصی تغییر می‌کند. این تغییرات به معنی افزودن راس و یال جدید به $V(q)$ یا حذف راس یا یال از آن است. به عنوان مثال، زمانی که ناظر q در شکل ۵—۳ از نقطه q به نقطه q' حرکت می‌کند، ساختار ترکیبیاتی $V(q)$ تغییر نمی‌کند و نیازی به بهنگام‌سازی ندارد.

درمقابل، زمانی‌که ما چندضلعی قابل دید دقیق را نگهداری می‌کنیم، باید مقدار دقیق قابل دید از یال‌های e_1 و e_2 در شکل ۵—۳ را مشخص کنیم. بنابراین، اگر ما بخواهیم چندضلعی قابل دید دقیق را در مهرهای زمانی[۴] متوالی داشته باشیم کافی است این یالها را شناسایی و در هر مهرزمانی مقدار دقیق این بخش‌ها را مشخص کنیم. در این‌صورت نمایش و رسم چندضلعی دقیق نیز با انجام اصلاحات لازم در اطراف همین بخش‌ها انجام می‌شود.

بسادگی می‌توان نشان داد که این تغییرات در مجاورت راس‌های انعکاسی قابل دید از q قرار دارند. راس v در شکل ۵—۵ نمونه‌ای از این راس‌ها است. در این شکل، فقط قسمت بالایی یال e که توسط امتداد پاره‌خط qv تعیین می‌شود از q قابل دید است. زمانی که q حرکت می‌کند، این بخش قابل دید برحسب جهت حرکت q، اضافه یا کم می‌شود. ناحیه‌هایی مانند A در این شکل که در مجاورت تغییر دید قرار دارند پستو نامیده می‌شوند. هر پستوی c با سه پارامتر راس، یال و پنجره آن مشخص می‌شود و به ترتیب با v_c، w_c و e_c نشان داده می‌شود. برای پستوی A در این شکل، راس، پنجره و یال آن به ترتیب v، $\overrightarrow{l}$ و e هستند.

[۴] Time-stamp



شکل ۵-۵: هنگام حرکت q فقط بخش قابل دید یال e از $V(q)$ تغییر می‌کند. A یک پستو با راس v، پنجره l و یال e است.

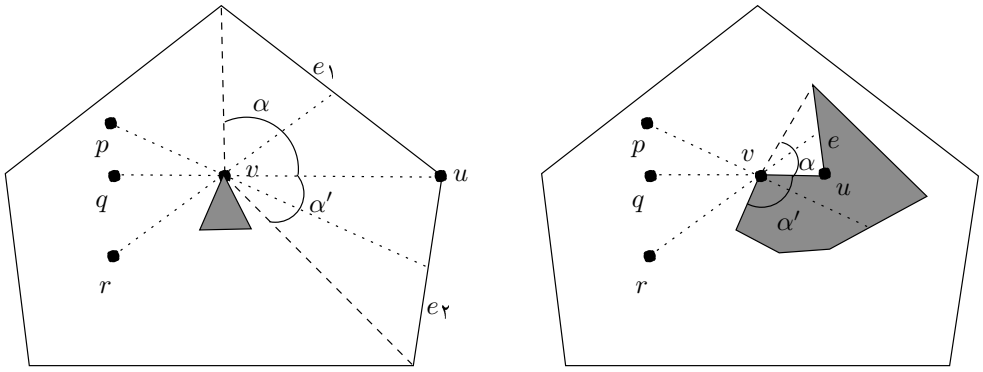
ایده اصلی الگوریتم ما برای نگهداری برخط مقدار دقیق $V(q)$ این است که در هنگام حرکت بتوانیم پستوها را در زمان بهینه شناسایی و بهنگام‌سازی کنیم. سپس در هر مهرزمانی، چندضلعی قابل دید دقیق با تعیین بخش قابل دید یال‌های پستوها در آن لحظه محاسبه می‌شود. پس از تعیین مقدار اولیه $V(q)$ با استفاده از الگوریتم‌های موجود، لیست پستوهای آن محاسبه و بصورت مرتب شده حول q نگهداری می‌شود. این لیست با $C(q)$ نشان داده می‌شود. این لیست مکان‌هایی از $V(q)$ را که هنگام حرکت q تغییر می‌کنند مشخص می‌کند. در ادامه، نحوه ساخت اولیه $C(q)$، روش تشخیص تغییرات احتمالی این لیست هنگام حرکت و روش اصلاح این لیست در هر مهرزمانی بیان می‌شوند.

۵-۵-۱ مرحله آغازین الگوریتم

قبل از شروع حرکت، $V(q)$ برای موقعیت اولیه q را براساس روش گفته شده در بخش ۵—۴ محاسبه می‌کنیم. طبق این روش، $V(q)$ شامل دنباله مرتب یال‌ها و راس‌های قابل دید از q است. با داشتن $V(q)$، $C(q)$ بصورت زیر محاسبه می‌شود. به ازای هر راس انعکاسی از $V(q)$، یک پستو به $C(q)$ اضافه می‌شود. این پستوها با یک پیمایش خطی روی $V(q)$ و تعیین رئوس انعکاسی مشخص می‌شوند. به ازای هر راس انعکاسی v_i، پستوی c_i متناظر با آن محاسبه می‌شود که راس، یال و پنجره این پستو به ترتیب برابر با v_i، e_i و $\overrightarrow{v_iv_i'}$ هستند. در اینجا، e_i یالی است که در دنباله $V(q)$ دقیقا بعد یا قبل از v_i قرار دارد و توسط امتداد پاره‌خط qv_i قطع می‌شود و v_i' محل این تقاطع است. واضح است که با داشتن $V(q)$ همه پستوها در زمان $O(|C(q)|)$ ساخته می‌شوند.

علاوه‌برآن، برای هر پنجره w_c، بازه $R_{v_c}(w_c)$ را که در برگیرنده پنجره w_c است نیز محاسبه می‌کنیم که برای پنجره هر پستو با جستجوی دودویی روی R_{v_c} در زمان $O(\log(|R_{v_c}|))$ قابل محاسبه است.

لم ۱۵ . پیچیدگی زمانی مرحله آغازین الگوریتم $O(|V(q)|\log n)$ است.



شکل ۵-۶: در بخش A (B)، هنگامی که ناظر از نقطه p به r یا از r به p حرکت می‌کند یک رخداد انتقال (افزودن-حذف) اتفاق می‌افتد.

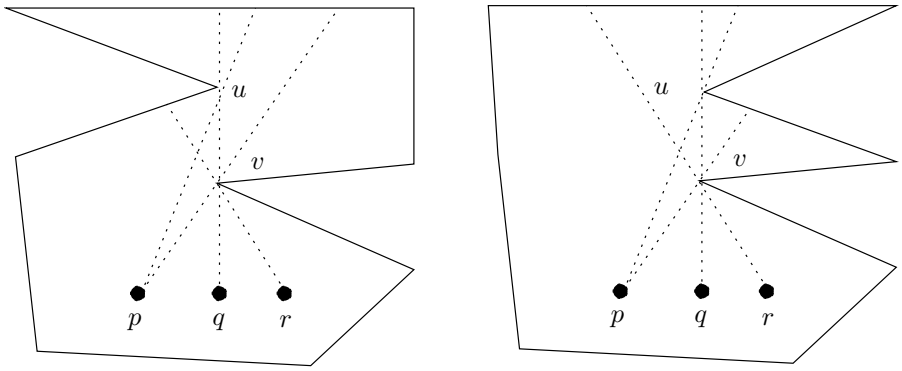
اثبات. طبق توضیحات بالا، $V(q)$ در زمان $O(|V(q)|\log n)$ محاسبه می‌شود. سپس، $C(q)$ در زمان $O(|V(q)|)$ محاسبه می‌شود و بازه‌های دربرگیرنده پنجره‌های پستوها نیز در زمان $O(|C(q)|\log n)$ بدست می‌آیند. □

۵-۵-۲ بهنگام‌سازی پیوسته دید

مقدار اولیه بدست آمده برای چندضلعی قابل دید دقیق در هنگام حرکت و در هر مهرزمانی باید بهنگام شود. ما از جهت و سرعت حرکت اطلاعی نداریم و در هر مهرزمانی موقعیت جدید q داده می‌شود و باید شکل جدید $V(q)$ را تهیه کنیم. برای این‌که بتوانیم این کار را در زمان مناسب انجام دهیم کافی‌است که لیست پستوهای آن لحظه را داشته باشیم. به همین منظور و در هر مهرزمانی که موقعیت جدید q ابلاغ می‌شود لیست $C(q)$ را بهنگام‌سازی و براساس این لیست مقدار دقیق $V(q)$ را مشخص می‌کنیم: به ازای هر پستوی c از $C(q)$، محل تقاطع e_c و امتداد qv_c را محاسبه می‌کنیم. این تقاطع مشخص کننده بخش قابل دید یال e_c در آن لحظه است.

این روش مادام که $C(q)$ ثابت است اعتبار دارد. ولی با گذشت زمان و حرکت بیشتر ناظر $C(q)$ تغییر می‌کند. بنابراین صحت این الگوریتم منوط به بهنگام‌سازی درست $c(q)$ در هر مهرزمانی است. برای این‌کار باید رخدادهایی را که باعث تغییر $C(q)$ می‌شوند شناسایی و به موقع پردازش کنیم. سه دسته رخداد باعث تغییر $V(q)$ می‌شوند که در شکل‌های ۵—۶ و ۵—۷ نشان داده شده‌اند.

دسته اول رخدادها که رخدادهای انتقالی نامیده می‌شوند در قسمت A از شکل ۵—۶ نشان داده شده است. در این شکل، زمانی که ناظر از نقطه p به نقطه r حرکت می‌کند قبل از رسیدن به نقطه q، e_1 یال یک پستو از $C(q)$ است و پس از گذشتن از q، e_2 این یال خواهد بود. زمانی که ناظر از r به p منتقل می‌شود نیز چنین رخدادی (بصورت وارون) پیش می‌آید. در یک رخداد



شکل ۵–۷: هنگامی که ناظر از نقطه p به r یا از r به p حرکت می‌کند یک رخداد تقسیم–ادغام اتفاق می‌افتد.

انتقالی، یک یال جدید به $V(q)$ اضافه می‌شود که جایگزین یال یکی از پستوها می‌شود یا اینکه یال یکی از پستوها از محدوده دید ناظر خارج می‌شود که باید از $V(q)$ حذف شود و یال این پستو برابر با یال مجاور آن در $V(q)$ خواهد شد.

دسته دوم رخدادها که در قسمت B از شکل ۵–۶ نشان داده شده است رخداد افزودن–حذف نامیده می‌شود. مطابق این شکل، هنگام حرکت ناظر از نقطه p به نقطه r، قبل از رسیدن به نقطه q هیچ پستوی موثری روی راس v وجود ندارد. یک پستو موثر است هرگاه پنجره آن درون صحنه قرار داشته باشد. ولی پس از عبور از نقطه q و در نقطه r یک پستوی موثر بر روی راس v وجود دارد. بنابراین، در نقطه q پستوی جدید باید به عنوان یک پستوی موثر به $C(q)$ اضافه شود. زمانی که ناظر از نقطه r به نقطه p حرکت کند این واقعه بصورت وارون اتفاق می‌افتد و در نقطه q باید پستوی متناظر با راس v به عنوان یک پستوی غیرموثر مشخص شود.

دسته سوم رخدادها، رخدادهای تقسیم–ادغام نامیده می‌شوند. در این رخدادها دو پستو با هم ادغام می‌شوند یا یک پستو به دو پستوی متمایز تقسیم می‌شود. شکل ۵–۷ این وضعیت‌ها را نشان می‌دهد. زمانی که ناظر روی نقطه q قرار دارد دو پستوی متمایز روی رئوس v و u دارد. با حرکت ناظر به سمت نقطه r این دو به هم نزدیک می‌شوند بطوری‌که در نقطه r فقط یک پستو روی راس v باقی می‌ماند. از طرف دیگر، زمانی که ناظر از r به p حرکت می‌کند این واقعه بصورت وارون اتفاق می‌افتد. ابتدا یک پستو روی راس v وجود دارد و در نقطه q به دو پستوی متمایز روی رئوس v و u تقسیم می‌شود.

بنابراین، در یک رخداد تقسیم–ادغام، براساس جهت حرکت ناظر یک عمل حذف یا افزودن پستو اتفاق می‌افتد که باید تغییرات مربوط در $C(q)$ و $V(q)$ اعمال شود.

برای اثبات صحت این روش باید نشان دهیم که سه دسته رخدادهای بالا همه تغییرات مربوط به پستوها را شامل می‌شوند.

قضیه ۱۴ . چنانچه تغییرات مربوط به سه دسته رخدادهای یاد شده در بالا بدرستی شناسایی و پردازش شوند، ساختار ترکیبیاتی $V(q)$ در طول حرکت معتبر خواهد ماند.

اثبات. برای اثبات نشان می‌دهیم که هر تغییر در ساختار ترکیبیاتی $V(q)$ متناظر با یک تغییر در $C(q)$ است. ساختار ترکیبیاتی $V(q)$ زمانی تغییر می‌کند که q از روی امتداد یک یال e از گراف قابلیت دید عبور کند. در این‌صورت یک راس و یال جدید به $V(q)$ افزوده می‌شود یا یک راس و یال از آن حذف می‌شود. فرض کنید e متناظر با راس‌های u و v است و u راسی است که اخیرا به $V(q)$ اضافه یا از آن حذف شده است (با این فرض که درست قبل از این رویداد (عبور q از روی e) مقدار درست $C(q)$ موجود است). بدیهی است v یک راس انعکاسی است و قبل از این رویداد، یک پستوی موثر یا غیرموثر متناظر با آن در $C(q)$ موجود بوده و پس از این رویداد یال این پستو تغییر کرده است. پس این تغییر در $V(q)$ متناظر با یک تغییر در $C(q)$ بوده است و اگر ما تغییرات $C(q)$ را درست شناسایی و پردازش کنیم می‌توانیم مقدار درست $V(q)$ را داشته باشیم.

با یک استدلال ساده می‌توان نتیجه گرفت که هر تغییر در $C(q)$ نیز متناظر با یکی از سه نوع رخداد بالاست. □

در بخش بعدی نحوه شناسایی و پردازش رخدادها را بیان می‌کنیم.

۵-۶ پردازش رخدادها

برخلاف سایر الگوریتم‌ها، مفهوم صف رخدادها در این‌جا وجود ندارد. به‌جای‌آن، در مهرهای زمانی مشخصی رخدادهای ممکن پردازش و عملیات لازم انجام می‌شود. در هر مهرزمانی، لیست پستوها براساس موقعیت جدید q بررسی می‌شود. با پیمایش این لیست، رخدادهایی که در فاصله مهرزمانی قبلی تا حالا اتفاق افتاده‌اند شناسایی می‌شوند. در مورد هر پستو، ممکن است بیش از یک رخداد شناسایی شود که باید به ترتیب وقوع آنها پردازش و اجرا شوند.

بنابراین، در خلال این پیمایش خطی روی لیست پستوها، هر پستو بررسی و در صورت نیاز رخداد مربوط به آن اجرا و تصحیحات لازم در $V(q)$ و $C(q)$ اعمال می‌شود. این کار برای یک پستو تا زمانی که هیچ رخداد پردازش نشده‌ای نداشته باشد تکرار می‌شود.

برای هر یک از این رخدادها، اصلاحات لازم در $V(q)$ و $C(q)$ مطابق آنچه که در بخش‌های بعدی خواهیم دید، انجام می‌شود. پس از پایان این پیمایش خطی، مقادیر درست $C(q)$ و $V(q)$ در مهرزمانی جاری بدست می‌آیند و چندضلعی قابل دید دقیق را می‌توان رسم کرد.

فرض کنید t مهرزمانی جاری و t' مهرزمانی قبلی است. در بخش‌های بعدی، یال و پنجره پستوی c را در مهرزمانی t' به ترتیب با e'_c و w'_c نشان می‌دهیم.

۵-۶-۱ پردازش رخدادهای انتقالی

یک رخداد انتقالی برای پستوی c زمانی اتفاق می‌افتد که w_c از روی یکی از یال‌های $R_{v_c}(w'_c)$ مثلا vu عبور کند و $e_{\overrightarrow{vu}}$ تهی باشد. برای تشخیص اینکه آیا در مهرزمانی جاری چنین رخدادی در مورد پستوی c اتفاق افتاده است کافی است امتداد جاری w_c را با اضلاع بازه $R_{v_c}(w'_c)$ مقایسه کنیم. در مهرزمانی t مقدار $R_{v_c}(w'_c)$ به ازای مهرزمانی قبلی را داریم. بنابراین، می‌توانیم این مقایسه را در زمان ثابت انجام دهیم.

پس از تشخیص وقوع چنین رخدادی، باید تغییرات لازم در $C(q)$ و $V(q)$ انجام شوند. همانند قسمت A از شکل ۵—٦، فرض کنید ناظر از نقطه p حرکت کرده است و هم‌اکنون در نقطه q قرار دارد. در اولین مهرزمانی پس از این لحظه این رخداد شناسایی می‌شود و اصلاحات مربوط به $C(q)$ و $V(q)$ بصورت زیر انجام می‌گیرند.

براساس نتایج مربوط به مهرزمانی قبلی، می‌دانیم که $e'_c = e_۲$ و فرض می‌کنیم که $R_{v_c}(w'_c) = \alpha'$. دراینصورت، $R_{v_c}(w_c) = \alpha$ که α بازه پس از α' در R_{v_c} در جهت پادساعت‌گرد است. همچنین، مقدار e_c را برابر با $e_۱$ قرار می‌دهیم که $e_۱$ یال دیگر متصل به راس u است. با داشتن گراف قابلیت دید تقویت شده، بدست آوردن این مقادیر و اصلاح مشخصات پستوی مربوط در زمان ثابت انجام می‌شود. بهنگام‌سازی $V(q)$ نیز معادل است با حذف $e_۲$ که با داشتن پیوندهای بین یال‌های پستوها به یال‌های چندضلعی قابل دید در زمان ثابت انجام‌پذیر است.

در مورد حرکت در جهت مخالف (شروع حرکت از نقطه r در قسمت A از شکل ۵—٦)، رخداد مربوط با تغییرات اندکی نسبت به حالت بالا انجام می‌شود. در این‌حالت، $e_۲$ باید به محل مناسب آن در $V(q)$ اضافه شود و مقدار $R_{v_c}(w_c)$ را برابر با α' قرار دهیم که α' بازه پیش از $R_{v_c}(w'_c)$ در جهت پادساعت‌گرد در R_{v_c} است.

۵-۶-۲ پردازش رخدادهای افزودن-حذف

یک رخداد افزودن-حذف برای پستوی c زمانی اتفاق می‌افتد که w_c از روی یکی از یال‌های $R_{v_c}(w'_c)$ مثلا vu عبور کند که vu یکی از یال‌های صحنه است. بنابراین، همانند روش تشخیص رخدادهای انتقالی و با همان پیچیدگی زمانی یعنی $O(۱)$ شناسایی می‌شوند و روش پردازش آنها نیز یکسان است. همانند قسمت B از شکل ۵—٦، فرض کنید که ناظر از نقطه p حرکت کرده و هم‌اکنون در نقطه q است. در این نقطه یک رخداد افزودن اتفاق می‌افتد: یال vu از $V(q)$ حذف می‌شود؛ پستوی c متناظر با راس v به عنوان یک پستوی موثر مشخص می‌شود؛ e_c برابر با e مقداردهی می‌شود که یال دیگر متصل به راس u است و $R_{v_c}(w_c)$ برابر با α مقداردهی می‌شود که بازه‌ی پس از $R_{v_c}(w'_c)$ در R_{v_c} در جهت پادساعت‌گرد است.

برای حرکت در جهت وارون (حرکت از نقطه r در قسمت B از شکل ۵—٦)، در نقطه q یک رخداد حذف اتفاق می‌افتد که تقریبا برعکس بالا پردازش می‌شود: vu در محل مناسب در

$V(q)$ درج می‌شود؛ $R_{v_c}(w_c)$ به α' مقداردهی می‌شود و پستوی c به عنوان ناموثر مشخص می‌شود. بسادگی قابل مشاهده است که با داشتن گراف قابلیت دید تقویت شده، هر رخداد افزودن–حذف در زمان ثابت پردازش می‌شود.

۵-۶-۳ پردازش رخدادهای تقسیم–ادغام

چنانچه در شکل ۵—۷ نشان داده شده است، یک رخداد تقسیم–ادغام روی پستوی c زمانی رخ می‌دهد که w_c از روی یکی از یال‌های $R_{v_c}(w'_c)$ مانند vu عبور کند و پنجره w_d مربوط پستوی راس انعکاسی u درست پس از این لحظه درون صحنه قرار گیرد. همانند رخدادهای قبلی، برای تشخیص رخدادهای تقسیم–ادغام کافی است امتداد w_c با یال‌های $R_{v_c}(w'_c)$ مقایسه شود و نیز وضعیت راس u و امتداد پنجره w_d بررسی شود که همگی در زمان ثابت قابل انجام هستند.

برای پردازش یک رخداد تقسیم، پستوی جدید d مربوط به راس u به لیست $C(q)$ و پس از پستوی c اضافه می‌شود. رئوس پستوهای c و d مشخص و به ترتیب برابر با v و u هستند. برای محاسبه یال‌های این پستوها و بازه‌های دربرگیرنده پنجره‌های آنها از اطلاعات موجود در گراف قابلیت دید تقویت شده استفاده می‌کنیم. بدون کاهش کلیت، فرض کنید v نسبت به u به ناظر نزدیک‌تر است. برای حالتی که در قسمت A از شکل ۵—۷ نشان داده شده است، e_c و e_d برابر است با $e_{\overrightarrow{vu}}$. برای حالتی که در قسمت B از شکل ۵—۷ نشان داده شده است، e_c برابر با یکی از یال‌های مجاور راس u است و e_d برابر است با $e_{\overrightarrow{vu}}$. از آنجا که vu یکی از یال‌های گراف قابلیت دید است، مقدار $e_{\overrightarrow{vu}}$ از این گراف قابل استخراج است. همچنین، $R_u(w_d)$ برابر است با $R_u(\overrightarrow{vu})$ که از گراف قابلیت دید تقویت شده بدست می‌آید و $R_v(w_c)$ برابر با یکی از بازه‌های مجاور $R_v(w'_c)$ است. بنابراین، مقادیر مربوط به پارامترهای پستوی جدید و پیشین در زمان ثابت قابل محاسبه هستند که نتیجه آن بهنگام‌سازی $C(q)$ در زمان ثابت هنگام پردازش یک رخداد تقسیم است. $V(q)$ نیز در زمان ثابت قابل بهنگام‌سازی است. برای این کار e_d، u و e_c پیش یا پس از موقعیت e'_c در $V(q)$ درج می‌شوند. چنانچه e_d یا e_c در $V(q)$ و در محل مورد نظر موجود باشند نیازی به افزودن مجدد آنها نیست.

چنانچه در شکل ۵—۷ نشان داده شده است، یک رخداد ادغام برای پستوهای c و d زمانی اتفاق می‌افتد که w_d و w_c همزمان از روی امتداد v_cv_d بگذرند و v_c و v_d در یک طرف ناظر باشند. خوشبختانه، زمانی که رئوس صحنه در موقعیت عمومی قرار دارند (هیچ سه راسی بر روی یک خط قرار ندارند) این رخداد فقط برای پستوهای مجاور هم می‌تواند پیش بیاید.

لم ۱۶ . فقط پستوهای مجاور هم در $C(q)$ با هم ادغام می‌شوند.

اثبات. زمانی که پنجره‌های دو پستو روی هم قرار می‌گیرند هیچ راس دیگری بین آنها نیست که از ناظر قابل دید باشد و در نتیجه رئوس این پستوها در $V(q)$ مجاور هستند. □

بنابراین، رخدادهای ادغام با بررسی وضعیت پستوهای مجاور در $C(q)$ تشخیص داده می‌شوند و این بررسی برای هر پستو از $C(q)$ به $O(1)$ زمان پردازش نیاز دارد. پس از تشخیص، هر رخداد ادغام بصورت زیر پردازش می‌شود. با این فرض که v نسبت به u نزدیکتر به ناظر است، پستوی d متناظر با راس u از $C(q)$ حذف می‌شود. برای حالتی که در قسمت A از شکل ۵—۷ نشان داده شده است، e_c برابر با e'_d مقداردهی می‌شود و برای حالتی که در قسمت B از این شکل نشان داده شده است، e_c برابر با یکی از یالهای مجاور u مقداردهی می‌شود. در هر دو حالت، مقدار $R_v(w_c)$ برابر با یکی از بازه‌های مجاور $R_v(w'_c)$ خواهد بود. برای بهنگام‌سازی $V(q)$، e'_c، u و e'_d از $V(q)$ حذف می‌شوند و بجای آنها e_c درج می‌شود. ممکن است e'_c و e'_d برابر باشند یا e_c در $V(q)$ و در محل مربوط وجود داشته باشد که در هنگام بهنگام‌سازی $V(q)$ این شرایط باید بطور مناسب در نظر گرفته شوند.

۵-۶-۴ تحلیل کارایی الگوریتم

لازم به ذکر است که در هر مهرزمانی ممکن است بر روی یک پستو چندین رخداد پردازش شود. این امر به این دلیل است که فاصله زمانی بین مهرهای‌زمانی می‌تواند طولانی باشد و در طول این مدت تغییرات دید مختلفی پیش بیاید. ما این رخدادها را براساس ترتیب وقوع آنها پردازش می‌کنیم تا نهایتا مقدار درست $C(q)$ و $V(q)$ بدست آید.

نتیجه زیر از بخش‌های قبلی بدست می‌آید.

لم ۱۷ . در هر مهرزمانی که مقدار دقیق $V(q)$ رسم می‌شود، زمان لازم برای بهنگام‌سازی $C(q)$ و $V(q)$ نسبت به تعداد رخدادهای تغییر دید که از مهرزمانی قبلی تاکنون رخ داده است خطی است.

با توجه به اینکه هر یک از این تغییرات باید در چندضلعی قابل دید مهرزمانی قبلی اعمال شود تا برابر با چندضلعی قابل دید در مهرزمانی جاری باشد، این زمان خطی برای پردازش رخدادها بهینه است.

چنانچه بازه‌های زمانی بین مهرهای‌زمانی به اندازه کافی کوچک باشد، می‌توانیم فرض کنیم که در هر مهرزمانی حداکثر یک رخداد روی هر پستوی $C(q)$ اتفاق می‌افتد. درنتیجه، زمان لازم برای بهنگام‌سازی و رسم چندضلعی قابل دید در هر مهرزمانی نسبت به مکان‌هایی از دید که طی این مهرزمانی تغییر کرده است خطی است. کارایی این الگوریتم بصورت زیر قابل بیان است.

قضیه ۱۵ . یک دامنه چندضلعی گونه را می‌توان به‌گونه‌ای پیش‌پردازش کرد که چندضلعی قابل دید دقیق برای ناظر نقطه‌ای q در مهرهای زمانی نزدیک به هم در زمان $O(|C(q)|)$ بهنگام‌سازی و نگهداری شود. در فاصله هر دو مهرزمانی، q با سرعت دلخواه و در جهت دلخواه می‌تواند حرکت کند.

قضیه ۱۶ . کل زمان پردازش برای نگهداری چندضلعی قابل دید دقیق ناظر نقطه‌ای متحرک q در یک مسیر داده شده برابر است با $O(k\overline{|C(q)|}+l)$ که k تعداد دفعاتی است که مقدار دقیق $V(q)$ تهیه می‌شود و l تعداد تغییرات ترکیبیاتی $V(q)$ و $\overline{|C(q)|}$ متوسط اندازه $C(q)$ در طول این مسیر است.

از نتایج بالا درمی‌یابیم که الگوریتم ارائه شده کاملا خروجی حساس[۵] است و نسبت به الگوریتم‌های موجود برتر است. پیچیدگی زمانی بهترین الگوریتم موجود برای شرایط گفته شده در قضیه ۱۶ برابر است با $O(k\overline{|V(q)|}+l\log n)$ که $\overline{|V(q)|}$ متوسط اندازه $V(q)$ در طول این مسیر است. همچنین، در این الگوریتم‌ها اگر ناظر تغییر مسیر دهد، $O(n\log n)$ زمان مورد نیاز است تا صف رخدادها مجددا ساخته شود، درحالیکه در روش ارائه شده نیازی به این پردازش اضافه نیست.

۵-۶-۵ مسائل مربوط به اعداد حقیقی

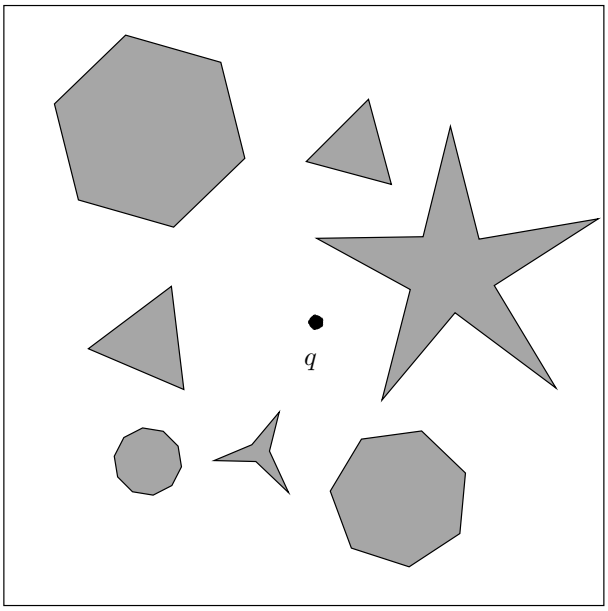
در مسائل هندسی، عملیات روی اعداد حقیقی یک چالش همیشگی است. این مساله از آنجا ناشی می‌شود که در حافظه کامپیوتری دقت اعداد حقیقی محدود است. خطای ناشی از حذف برخی ارقام اعداد حقیقی می‌تواند اثرات پیش‌بینی نشده‌ای را در یک الگوریتم ایجاد کند.

در الگوریتم‌های مبتنی بر رخداد یک صف از رخدادهای آینده نگهداری می‌شود که زمان بروز هر یک با توجه به دقت نگهداری اعداد اعشاری ممکن است دقیق نباشد. بنابراین، وقتی ما در زمان پیش‌بینی شده برای یک رخداد قرار می‌گیریم، وضعیت صحنه ممکن است اندکی با آنچه پیش‌بینی می‌کردیم متفاوت باشد. به عنوان مثال، ما می‌خواهیم یک رخداد را در زمان t که سه نقطه هم‌راستا هستند پردازش کنیم. با توجه به دقت موجود برای اعداد حقیقی زمان این رخداد را برابر با t' ثبت می‌کنیم که دقیقا برابر با t نیست. دراینصورت، در زمان t' که رخداد مربوط پردازش می‌شود دیگر سه نقطه مزبور هم‌راستا نیستند و این مساله ممکن است ناسازگاریهایی در الگوریتم بوجود آورد. این چالش در همه الگوریتم‌های هندسی مد نظر است و باعث پیچیدگی پیاده‌سازی این الگوریتم‌ها در کاربردهای واقعی می‌شود.

به عنوان یک مزیت دیگر برای الگوریتم پیشنهادی، ما در الگوریتم ارائه شده صف رخداد تشکیل نمی‌دهیم. درعوض، همه رخدادهایی که از مهرزمانی قبلی تاکنون اتفاق افتاده‌اند شناسایی و پردازش می‌شوند. بنابراین همه رخدادها و با ترتیب درست شناسایی و پردازش می‌شوند و این‌کار بسادگی قابل پیاده‌سازی در کاربردهای واقعی است.

---

[۵] Output Sensitive



شکل ۵–۸: یک صحنه مجازی.

۵–۷ نتایج پیاده‌سازی

با توجه به دید کاربردی الگوریتم مربوط به نگهداری چندضلعی قابل دید دقیق، کارایی این الگوریتم را در عمل بررسی کرده‌ایم که نتایج آن در ادامه بیان می‌شود. ما هم از چندین محیط واقعی و هم از محیط‌های مجازی به عنوان داده ورودی برای این ارزیابی استفاده کردیم. داده‌های مربوط به محیط‌های واقعی دارای اندازه محدود بودند و فقط برای مقایسه با داده‌های تصادفی تهیه شدند. با توجه به نزدیکی نتایج بدست آمده، در اینجا فقط نحوه چیدمان داده‌های تولید شده تصادفی بیان می‌شود.

مرز بیرونی صحنه از یک مربع تشکیل می‌شود و موانع درون آن بصورت k–ضلعی‌های منتظم یا k–ستاره‌های منتظم هستند و ناظر q در مرکز این مربع قرار دارد. موانع بصورت یکنواخت درون مربع بیرونی توزیع می‌شوند با این شرط که همدیگر را قطع نکنند. در شکل ۵—۸ یک مثال آمده است.

برای مقایسه صحنه‌های مختلف، پارامترهای زیر را برای یک صحنه S تعریف می‌کنیم.

- **چگالی** (d): چگالی یک صحنه بیانگر نسبت فضای پوشیده شده توسط موانع است و به صورت نسبت مساحت موانع به مساحت مرز بیرونی تعریف می‌شود.
- **پیچیدگی موانع** (n_h): پیچیدگی موانع یک صحنه برابر با میانگین تعداد راس‌های موانع است و بطور رسمی برابر با تعداد رئوس کل موانع تقسیم‌بر تعداد موانع است.

- **ریزدانگی مهرزمانی** (t): ریزدانگی مهرزمانی بصورت نسبت فاصله طی‌شده توسط ناظر بین دو مهرزمانی متوالی به طول مربع بیرونی است.
- **متوسط تعداد رخدادها** (n_e): متوسط تعداد رخدادها برابر با میانگین تعداد رخدادهایی است که در هر مهرزمانی باید پردازش شوند. ما این پارامتر را برای اولین مهرزمانی پس از حرکت ناظر از موقعیت اولیه آن و در امتداد قطرهای مربع بیرونی محاسبه می‌کنیم.
- **نرخ تعداد رخداد** (c): نرخ تعداد رخدادها را بصورت $\frac{|C(q)|}{|V(q)|}$ برای موقعیت اولیه q تعریف می‌کنیم.

نتایج پیاده‌سازی در شکل‌های ۵—۹، ۵—۱۰ و ۵—۱۱ نشان داده شده‌اند. برای بررسی رابطه بین n_e و t صحنه‌های مختلفی با ۱۰۰۰ مانع و برای $d \in \{۰.۰۰۰۰۱, ۰.۰۰۰۰۲, \ldots, ۰.۰۰۰۱\}$ و $n_h \in \{۵, ۱۰\}$ ساخته شدند. در شکل ۵—۹ نتیجه کار دیده می‌شود.

سپس برای مقدار ثابت $t = ۰.۰۰۰۱$ رابطه بین n_e و h بررسی شد. برای این کار، صحنه‌هایی با $h \in \{۱۰۰۰, ۲۰۰۰, \ldots, ۱۰۰۰۰\}$، $d \in \{۰.۰۰۰۰۱, ۰.۰۰۰۰۲, \ldots, ۰.۰۰۰۱\}$ و $n_h \in \{۵, ۱۰\}$ تهیه شدند. نتیجه کار در شکل ۵—۱۰ نشان داده شده است.

نهایتا، رابطه بین c با سایر پارمترهای صحنه بررسی شد. از نتایج کار دریافتیم که n_h و تعداد رئوس انعکاسی موانع مهمترین پارامترهای موثر بر c هستند. در مورد صحنه‌های مجازی ما، دو نوع مانع وجود دارد که متوسط تعداد رئوس انعکاسی آنها ضریب ثابتی از n_h است. بنابراین، ما فقط رابطه بین c و n_h را در شکل ۵—۱۱ نشان داده‌ایم.

از مجموعه‌ی این نمودارها نتیجه‌گیری‌های ارزشمند زیر استنباط می‌شوند.

۱) رابطه بین n_e و t خطی است: برای مقادیر داده شده d، h و n_h، با رشد t مقدار n_e بصورت خطی بزرگ می‌شود.

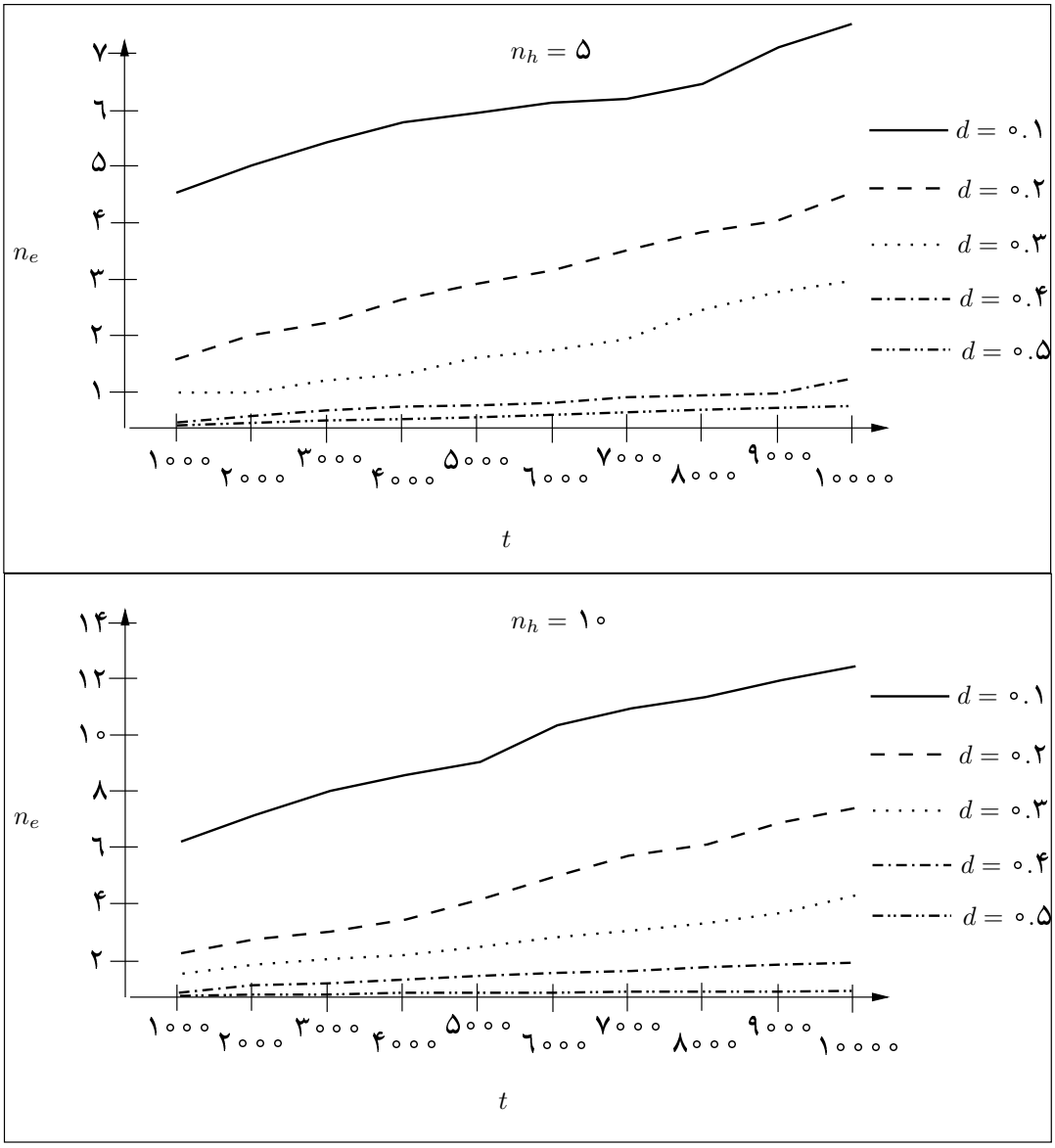
۲) رابطه بین n_e و h خطی است: برای مقادیر داده شده d، t و n_h، با رشد h مقدار n_e بصورت خطی بزرگ می‌شود.

۳) رابطه بین n_e و d فراتر از یک رابطه خطی است: برای مقادیر داده شده t، h و n_h، با رشد d مقدار n_e بصورت فراخطی[۶] بزرگ می‌شود.

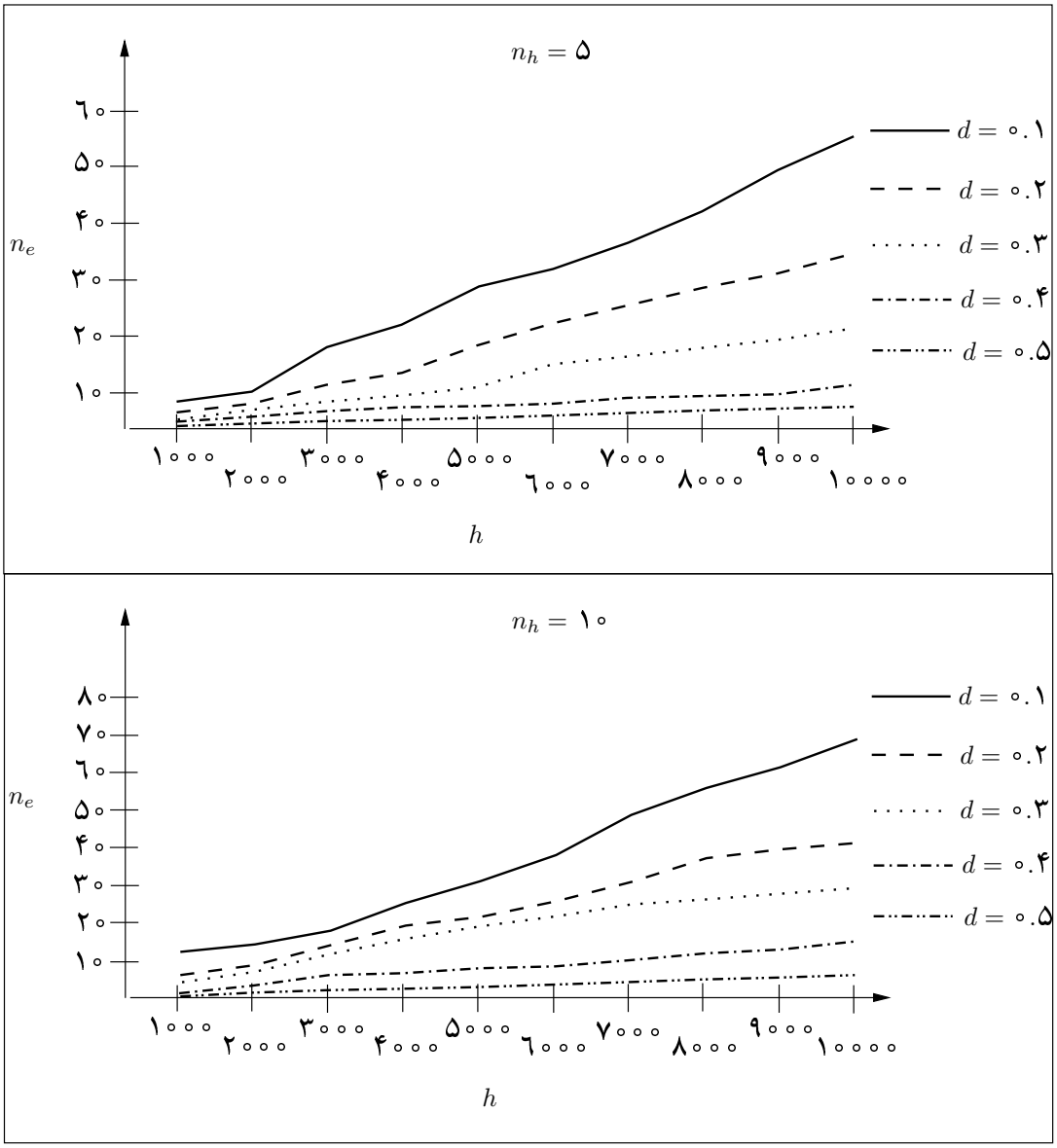
۴) رابطه بین n_e و n_h خطی است: برای مقادیر داده شده d، h و t، با رشد n_h مقدار n_e بصورت خطی بزرگ می‌شود.

۵) در یک صحنه متراکم ($d = ۰.۵$) مقدار n_e تقریبا ثابت است. واضح است که n_e به t بستگی دارد ولی برای مقادیر کوچک t ($t < \frac{۱}{۱۰h}$)، برای هر دو مقدار $n_h = ۵$ و $n_h = ۱۰$، مقدار n_e

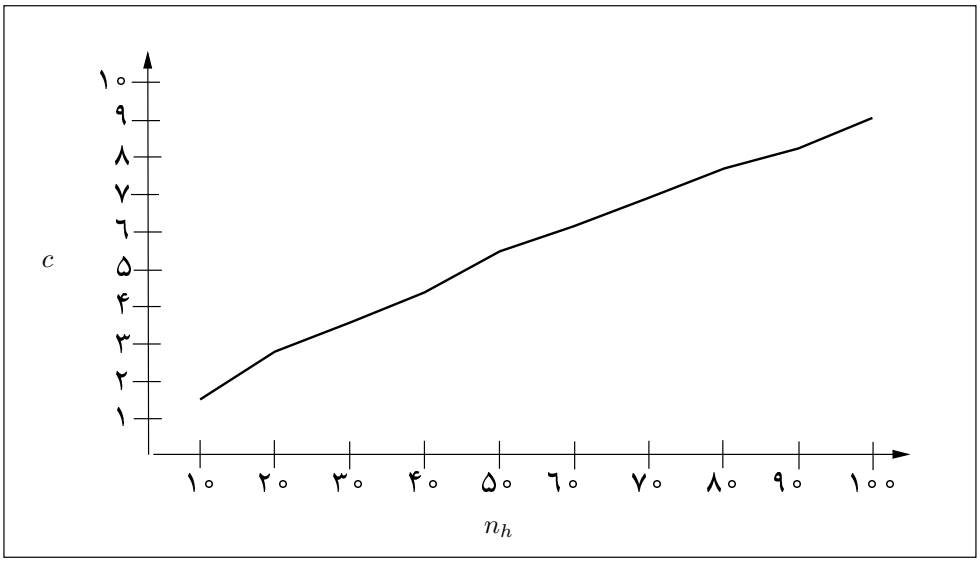
[۶] Super Linear



شکل ۵-۹: رابطه بین n_e و t برای $h = ۱۰۰۰$.



شکل ۵–۱۰: رابطه بین n_e و h برای $t = ۰.۰۰۰۱$.



شکل ۵-۱۱: رابطه بین c و n_h.

کمتر از یک است. علاوه‌برآن، در یک صحنه تُنُک ($d = ۰.۱$) با افزایش n_h مقدار n_e بسرعت رشد می‌کند.

۶) رابطه بین c و n_h خطی است: بدون توجه به مقادیر d، t و h، با رشد n_h مقدار c بصورت خطی بزرگ می‌شود.

از ملاحظات بالا نتیجه می‌گیریم که کاهش پیچیدگی زمانی محاسبه و ترسیم $V(q)$ از $|V(q)|$ به $|C(q)|$ یک بهبود ارزشمند است که در این الگوریتم بدست آمده است. همچنین، عدم نیاز به پردازشی جدا از فرایند ترسیم $V(q)$ نتیجه قابل قبول دیگری است که در این الگوریتم به آن رسیدیم.

۵-۸ نتیجه‌گیری

در این فصل مساله نگهداری فضای قابل دید از یک ناظر نقطه‌ای متحرک در صفحه بررسی شد. ابتدا، هدف نگهداری ساختار ترکیبیاتی $V(q)$ بود که فقط در زمان‌های خاصی که این ساختار تغییر می‌کند تغییرات لازم در آن اعمال می‌شود. در مقایسه با روش‌های قبلی، الگوریتم ارائه شده برای این مساله تعداد رخدادهای کمتری را در زمان مناسب (لگاریتمی) پردازش می‌کند. البته زمان پیش‌پردازش این روش نسبت به روش‌های قبلی بیشتر است.

مسائلی که در ادامه این الگوریتم قابل مطالعه و بررسی هستند شامل کاهش زمان پیش‌پردازش، تعمیم این روش به محیط‌های پویا با موانع متحرک، استفاده از این روش برای محاسبه چندضلعی قابل دید از یک پاره‌خط و نیز نگهداری چندضلعی قابل دید از یک پاره‌خط است.

مساله دیگری که دراین فصل معرفی و حل شد، نگهداری دقیق و نه ساختار ترکیبیاتی چندضلعی قابل دید بود. برای این مساله الگوریتمی با پیش پردازش $O(n^2)$ و زمان پرس‌وجوی بهینه ارائه شد. این روش بسادگی قابل پیاده‌سازی و بطور مستقیم قابل استفاده در برنامه‌های کاربردی است. نتایج پیاده‌سازی این الگوریتم موید کارایی آن در عمل است. در این روش محدودیتی در سرعت و جهت حرکت وجود ندارد و برخلاف الگوریتم‌های موجود، تغییر آنها نیازمند پردازش اضافی نیست.

کاهش زمان پیش‌پردازش این روش، بکارگیری این روش در محیط‌های پویا، گونه‌های دیگر ناظر و نیز محیط‌های سه‌بعدی مسیرهای توسعه این الگوریتم هستند.

فصل ۶

نگهداری چندضلعی قابل دید ناظر پاره‌خطی متحرک

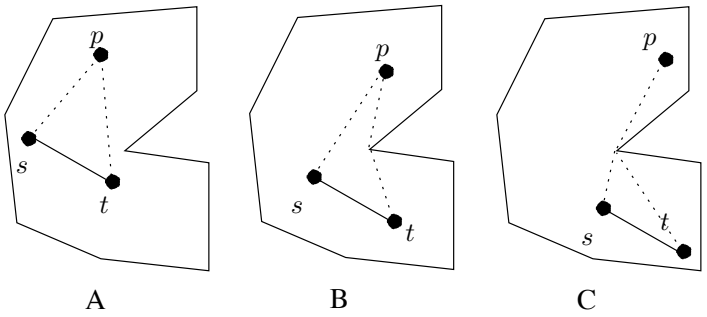
در بخش ۲—۴ الگوریتم‌هایی بهینه برای تعیین چندضلعی قابل دید ضعیف و قوی یک پاره‌خط در چندضلعی‌های ساده و حفره‌دار معرفی شدند. در این فصل، ابتدا روش حل این مساله را با استفاده از نتایج مربوط به نگهداری چندضلعی قابل دید برای یک ناظر نقطه‌ای بیان می‌کنیم.

سپس، مساله نگهداری چندضلعی قابل دید ضعیف و قوی یک ناظر پاره‌خطی متحرک در چندضلعی ساده را بررسی می‌کنیم. برای این مساله از درخت کوتاه‌ترین مسیر استفاده می‌کنیم که یک داده‌ساختار با اندازه خطی نسبت به تعداد رئوس چندضلعی است. پس از محاسبه مقدار اولیه چندضلعی قابل دید قوی و ضعیف یک پاره‌خط، تغییرات این دیدها را در زمان لگاریتمی اعمال می‌کنیم [۸۴].

۶-۱ مقدمه

در این فصل چندضلعی قابل دید ضعیف و قوی پاره‌خط st را به ترتیب با $SV(st)$ و $WV(st)$ نشان می‌دهیم. چنانچه در بخش ۲—۴ گفته شده، این مسائل در حالت ایستا و بصورت مستقیم حل شده‌اند. همچنین در فصل ۵ روش‌های نگهداری چندضلعی قابل دید یک ناظر نقطه‌ای متحرک بیان شد. با توجه به اینکه چندضلعی‌های قابل دید از یک پاره‌خط وابسته به تغییرات دید یک نقطه متحرک در طول این پاره‌خط است، می‌توان مساله محاسبه چندضلعی‌های قابل دید از یک پاره‌خط را با این روش‌ها نیز حل کرد.

بدین منظور، مقدار $V(s)$ به عنوان مقدار اولیه $WV(st)$ و $SV(st)$ در نظر گرفته می‌شود.



شکل ۶-۱: وضعیت‌های مختلف قابلیت دید نقطه و پاره‌خط در چندضلعی ساده.

سپس، برای ناظر نقطه‌ای متحرک از s به t تغییرات $V(s)$ محاسبه می‌شود و اصلاحات لازم در $WV(st)$ و $SV(st)$ صورت می‌گیرد.

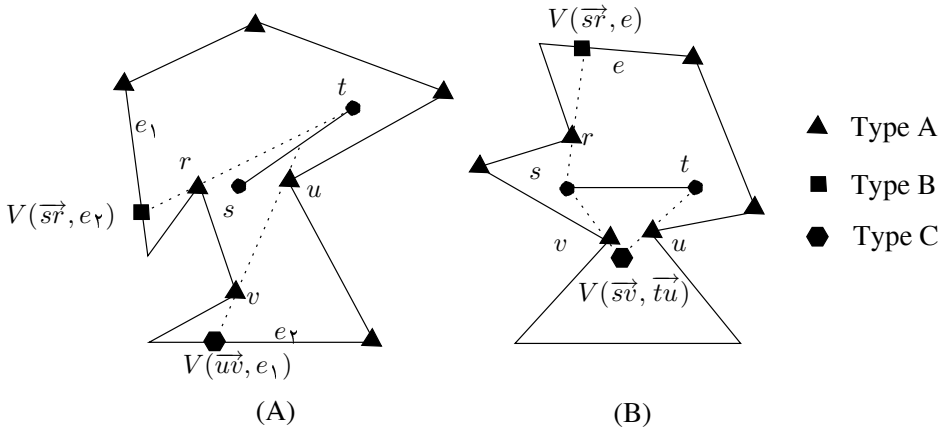
اما مساله‌ای که در این فصل به آن می‌پردازیم نگهداری $WV(st)$ و $SV(st)$ هنگام حرکت st است. ما این مساله را فقط برای چندضلعی‌های ساده حل می‌کنیم. برای حل این مساله، از درخت کوتاه‌ترین مسیر در چندضلعی‌های ساده استفاده می‌کنیم [۱۱]. براین اساس الگوریتمی برای نگهداری $WV(st)$ و $SV(st)$ برای پاره‌خط متحرک st درون یک چندضلعی ساده n راسی ارائه می‌دهیم که زمان و حافظه پیش‌پردازش آن به ترتیب برابر با $O(n \log n)$ و $O(n)$ است. سپس، هر تغییر در ساختار ترکیبیاتی $WV(st)$ و $SV(st)$ در زمان $O(\log^2 n)$ انجام می‌شود. درصورتی که مسیر حرکت st مشخص باشد این تغییرات در زمان بهینه $O(\log n)$ اعمال می‌شوند.

برای سادگی مساله، فرض می‌کنیم رئوس چندضلعی صحنه در موقعیت عمومی قرار دارند و هیچ سه راسی هم‌راستا نیستند. همچنین فرض می‌کنیم موقعیت پاره‌خط ناظر براساس یک تابع جبری با توان ثابت برحسب زمان قابل بیان است.

در ادامه، در بخش ٦—٢ خصوصیات و ویژگی‌های دید از یک پاره‌خط در چندضلعی ساده بیان می‌شود. در بخش ٦—٣ نحوه محاسبه دید اولیه و ساختارهای مورد نیاز برای بهنگام‌سازی دید در هنگام حرکت بیان می‌شود. در بخش ٦—۴ نحوه نگهداری دید هنگام حرکت و در بخش ٦—۵ تحلیل الگوریتم می‌آید.

۶-۲ ویژگی‌های دید از یک پاره‌خط

برای ناظر نقطه‌ای p و پاره‌خط st در چندضلعی ساده $\mathcal{P}$ سه حالت وجود دارد: p همه نقاط st را می‌بیند؛ p یک قطعه از st را می‌بیند؛ p هیچ نقطه‌ای از st را نمی‌بیند. این حالت‌ها در شکل ٦—١ نشان داده شده‌اند.



شکل ۶-۲: انواع رئوس در چندضلعی قابل دید قوی و ضعیف.

بخش قابل دید با رسم کوتاه‌ترین مسیر از p به رئوس s و t بدست می‌آید. ناحیه‌ای که بین دو امتداد متصل به p در این مسیرها قرار دارد گوشه دید[۱] نقطه p نسبت به پاره‌خط st نامیده و با $VC(p, st)$ نشان داده می‌شود. طبق قسمت A از شکل ٦—١، اگر st بطور کامل درون $VC(p, st)$ قرار داشته باشد آنگاه نقطه p جزء $SV(st)$ است. اگر st گوشه دید $VC(p, st)$ را قطع کند (قسمت B از شکل ٦—١) آنگاه نقطه p جزء $WV(st)$ است. در غیر این‌صورت (قسمت C از شکل ٦—١)، p نسبت به st قابل دید نیست.

بنابراین با داشتن کوتاه‌ترین مسیر از دو انتهای پاره‌خط st به همه رئوس چندضلعی $\mathcal{P}$، می‌توانیم مجموعه راس‌هایی را که بصورت قوی یا ضعیف از st قابل دید هستند را مشخص کنیم. با داشتن این راس‌ها می‌توانیم $SV(st)$ و $WV(st)$ را محاسبه کنیم.

بسادگی می‌توان نشان داد که در یک چندضلعی ساده، فقط یک قطعه به هم پیوسته (در صورت وجود) از هر ضلع در چندضلعی قابل دید قوی و ضعیف یک پاره‌خط وجود دارد. همانطور که توسط گیباس[۲] و همکاران [۶۲] نشان داده شده است، راس‌های $WV(st)$ سه نوع هستند که در قسمت A از شکل ٦—٢ نشان داده شده‌اند.

- نوع A: راس‌هایی از $\mathcal{P}$ که از s یا t قابل دید هستند.
- نوع B: راس‌هایی که با امتداد دادن پاره‌خط متصل کننده یک انتهای x از st و یک راس قابل دید v از $\mathcal{P}$ بدست می‌آیند. اگر e ضلع قطع شده در امتداد شعاع $\overrightarrow{xv}$ باشد، راس بوجود آمده را با $V(\overrightarrow{xv}, e)$ نشان می‌دهیم.

---

[۱] Visible cone

[۲] Guibas

- نوع C: راس‌هایی که از امتداد دادن پاره‌خط متصل کننده دو راس قابل دید u و v، زمانی که این امتداد پاره‌خط st را قطع می‌کند، بدست می‌آیند و با $V(\overrightarrow{uv}, e)$ نشان داده می‌شوند که e یال قطع شده توسط این امتداد است.

راس‌هایی که در $SV(st)$ ظاهر می‌شوند نیز سه دسته‌اند که در قسمت B از شکل ٦—٢ نشان داده شده‌اند.

- نوع A: راس‌هایی از $\mathcal{P}$ که از s و t قابل دید هستند.
- نوع B: راس‌هایی که با امتداد دادن پاره‌خط متصل کننده یک انتهای x از st و یک راس قابل دید v از $\mathcal{P}$ بدست می‌آیند. اگر e ضلع قطع شده در امتداد شعاع $\overrightarrow{xv}$ باشد، راس بوجود آمده را با $V(\overrightarrow{xv}, e)$ نشان می‌دهیم.
- نوع C: راس‌هایی که از محل تقاطع دو امتداد که نقاط s و t را به ترتیب به دو راس قابل دید v و u وصل می‌کنند بدست می‌آیند که با $V(\overrightarrow{sv}, \overrightarrow{tu})$ نشان داده می‌شوند.

۶-۳ محاسبه ساختارها و دید اولیه

برای محاسبه و نگهداری چندضلعی‌های قابل دید قوی و ضعیف یک پاره‌خط از داده‌ساختار درخت کوتاه‌ترین مسیر استفاده می‌کنیم. درخت کوتاه‌ترین مسیر از یک نقطه p واقع در چندضلعی ساده $\mathcal{P}$ با $SPT(p)$ نشان داده می‌شود و بدین صورت تعریف می‌شود که ریشه درخت، نقطه p و گره‌های آن راس‌های $\mathcal{P}$ است. یال‌های این درخت متناظر با همه پاره‌خط‌هایی هستند که در کوتاه‌ترین مسیر راس‌های $\mathcal{P}$ به p ظاهر می‌شوند.

ما می‌توانیم درخت‌های کوتاه‌ترین مسیر $SPT(s)$ و $SPT(t)$ را در زمان $O(n \log n)$ بسازیم. با یک پیمایش خطی روی این درخت‌ها، $WV(st)$ را می‌توانیم در زمان $O(n)$ محاسبه کنیم [۶۲]. با روشی مشابه، $SV(st)$ نیز در زمان خطی قابل محاسبه است که جزییات آن شبیه روش ارائه شده در [۶۲] است.

برای تشخیص تغییرات $WV(st)$ و $SV(st)$ می‌توانیم تغییرات $SPT(s)$ و $SPT(t)$ را هنگام حرکت st دنبال کنیم. زمانی که نقطه p در چندضلعی ساده $\mathcal{P}$ حرکت می‌کند، $SPT(p)$ فقط در لحظات خاصی از حرکت تغییر می‌کند. این ایده برای محاسبه درخت کوتاه‌ترین مسیر از همه رئوس $\mathcal{P}$ و نیز گراف قابلیت دید $\mathcal{P}$ استفاده شده است [۷۲]. زمانی که $SPT(s)$ یا $SPT(t)$ تغییر می‌کند ساختار ترکیبیاتی $WV(st)$ یا $SV(st)$ نیز تغییر می‌کند. در [۱۱] روشی برای پیش‌بینی تغییرات $SPT(q)$ برای نقطه متحرک q ارائه شده است. در این روش، اگر q بصورت خط مستقیم حرکت کند هر تغییر $SPT(q)$ در زمان $O(\log(|V(q)|))$ پردازش می‌شود. برای حالتی که q می‌تواند تغییر جهت دهد هر تغییر $SPT(q)$ در زمان $O(\log^2(|V(q)|))$ پردازش می‌شود. ما نیز می‌توانیم

تغییرات $SPT(s)$ و $SPT(t)$ را به همین روش پیگیری و اعمال کنیم و در طول حرکت بجای اینکه هر بار $WV(st)$ و $SV(st)$ را از اول بسازیم تغییرات مربوط را در هر تغییر $SPT(s)$ و $SPT(t)$ در آنها اعمال کنیم.

در هنگام حرکت، $WV(st)$ و $SV(st)$ بصورت پیوسته تغییر می‌کنند ولی $SPT(s)$ و $SPT(t)$ بصورت گسسته و در مواقع مشخصی تغییر می‌کنند. برای حل این مشکل، فقط ساختار ترکیبیاتی $WV(st)$ و $SV(st)$ را نگهداری می‌کنیم که تغییرات آن متناظر با تغییرات $SPT(s)$ و $SPT(t)$ است. ساختار ترکیبیاتی $WV(st)$ و $SV(st)$ را بصورت دنباله راس‌های تشکیل دهنده این چندضلعی درنظر می‌گیریم. برخی از این راس‌ها همان راس‌های $\mathcal{P}$ هستند و برخی دیگر بر روی یال‌های $\mathcal{P}$ یا درون آن قرار دارند که همان راس‌های نوع B و C هستند که در بخش ٦—٢ تعریف شدند. با توجه به این‌که محل دقیق راس‌های اخیر با حرکت st بصورت پیوسته تغییر می‌کند، آنها را براساس راسها و یال‌های تعیین‌کننده آنها نگهداری می‌کنیم. با داشتن این اطلاعات، می‌توانیم مقدارهای دقیق $WS(st)$ و $SV(st)$ را با یک پیمایش خطی محاسبه کنیم.

لم ۱۸ . با استفاده از دنباله‌های رئوس $WV(st)$، مقدار دقیق $WV(st)$ را می‌توان در زمان $O(|WV(st)|)$ محاسبه کرد.

اثبات. طبق تعریف مربوط به دنباله راس‌های $WV(st)$، فقط لازم است که محل دقیق راس‌های نوع B و C را که در این دنباله وجود دارند مشخص کنیم. برای راس نوع B مثلا $V(\overrightarrow{xv}, e)$، اطلاعات مربوط به خط‌های گذرنده از xv و یال e را داریم. بنابراین می‌توانیم موقعیت دقیق آن را در زمان ثابت مشخص کنیم. برای یک راس نوع C مثلا $V(\overrightarrow{uv}, e)$ نیز با توجه به مشخص بودن uv و e می‌توانیم موقعیت آن‌را در زمان ثابت محاسبه کنیم. بنابراین، محل دقیق همه رئوس $WV(st)$ با یک پیمایش خطی روی دنباله $WV(st)$ بدست می‌آید. □

لم ۱۹ . با استفاده از دنباله‌های رئوس $SV(st)$، مقدار دقیق $SV(st)$ را می‌توان در زمان $O(|SV(st)|)$ محاسبه کرد.

اثبات. همانند اثبات مربوط به $WV(st)$، فقط لازم است که محل دقیق راس‌های نوع B و C را در دنباله $SV(st)$ مشخص کنیم. برای راس نوع B مثلا $V(\overrightarrow{xv}, e)$، اطلاعات مربوط به خط‌های گذرنده از xv و یال e را داریم. بنابراین می‌توانیم موقعیت دقیق آن را در زمان ثابت مشخص کنیم. برای یک راس نوع C مثلا $V(\overrightarrow{sv}, \overrightarrow{tu})$ نیز با توجه به مشخص بودن sv و tu می‌توانیم موقعیت آن‌را در زمان ثابت محاسبه کنیم. بنابراین، محل دقیق همه رئوس $SV(st)$ با یک پیمایش خطی روی دنباله $SV(st)$ بدست می‌آید. □

بنابراین، در ادامه $WV(st)$ و $SV(st)$ را به معنای دنباله رئوس در نظر می‌گیریم و روش بهنگام‌سازی آنها را بیان می‌کنیم.

۶-۴ نگهداری و بهنگام‌سازی دید هنگام حرکت

نشان می‌دهیم که برای اعمال تغییرات $WV(st)$ و $SV(st)$ کافی است تغییرات مربوط به $SPT(s)$ و $SPT(t)$ را دنبال کنیم.

لم ۲۰ . $WV(st)$ مربوط به پاره‌خط متحرک st در یک چندضلعی ساده فقط زمانی تغییر می‌کند که $SPT(s)$ یا $SPT(t)$ تغییر کند.

اثبات. واضح است که مجموعه رئوس نوع A فقط زمانی تغییر می‌کند که مجموعه رئوس سطح اول در یکی از درخت‌های کوتاه‌ترین مسیر s یا t تغییر کند. زمانی که یک راس نوع B یا C در $WV(st)$ ظاهر می‌شود، یک راس $\mathcal{P}$ از $V(s)$ یا $V(t)$ حذف شده است. همچنین غایب شدن یک راس نوع B یا C از $WV(st)$ به دلیل ظاهر شدن یک راس $\mathcal{P}$ در $V(s)$ یا $V(t)$ است. بنابراین، تغییر این نوع راس‌ها و در نتیجه تغییر $WV(st)$ فقط به هنگام تغییر در $SPT(s)$ یا $SPT(t)$ اتفاق می‌افتد. □

لم ۲۱ . $SV(st)$ مربوط به پاره‌خط متحرک st در یک چندضلعی ساده فقط زمانی تغییر می‌کند که $SPT(s)$ یا $SPT(t)$ تغییر کند.

اثبات. برای $SV(st)$، در برخی موارد دو راس نوع B بر اثر حرکت st تبدیل به یک راس نوع C می‌شوند یا اینکه یک راس نوع C به دو راس نوع B تقسیم می‌شود. برای سایر حالت‌ها با استدلالی مشابه آنچه در مورد $WV(st)$ گفته شد می‌توان لم را اثبات کرد. در مورد این دو حالت تغییری در $SPT(s)$ یا $SPT(t)$ رخ نمی‌دهد در حالی‌که $SV(st)$ تغییر می‌کند. بنابراین، فقط دنبال کردن تغییرات $SPT(s)$ و $SPT(t)$ برای بهنگام‌سازی $SV(st)$ کافی نیست. این مساله را با یک بررسی اضافی هنگام محاسبه مقدار دقیق $SV(st)$ (علاوه بر پیمایش خطی آن) انجام می‌دهیم. فرض کنید دو راس نوع B مانند $V(\overrightarrow{tu}, e)$ و $V(\overrightarrow{sv}, e)$ در هنگام حرکت با هم ادغام شده‌اند و یک راس نوع C مانند $V(\overrightarrow{sv}, \overrightarrow{tu})$ بوجود آورده‌اند و ما متوجه این تغییر برای بهنگام‌سازی $SV(st)$ نشده‌ایم.

حال اگر بخواهیم مقدار دقیق $SV(st)$ را با یک‌بار پیمایش دنباله یال‌های $SV(st)$ محاسبه کنیم متوجه می‌شویم که موقعیت دقیق راس‌های متناظر با $V(\overrightarrow{tu}, e)$ و $V(\overrightarrow{sv}, e)$ روی یال e مطابق با ترتیب این راس‌ها در $SV(st)$ نیست. همین نکته کافی‌است تا از رخداد پیش آمده برای این دو راس مطلع شویم و آن‌را اصلاح کنیم.

بنابراین، بااینکه فقط با دنبال کردن تغییرات $SPT(s)$ و $SPT(t)$ ممکن است برخی تغییرات $SV(st)$ را متوجه نشویم، ولی این تغییرات هنگام محاسبه مقدار دقیق $SV(st)$ یا در تغییرات بعدی شناسایی و بدون افزایش زمان این فرایند اصلاحات لازم در $SV(st)$ انجام می‌شود. □

بنابراین، زمان حرکت st، تغییرات مربوط به $SPT(s)$ و $SPT(t)$ شناسایی و پردازش می‌شوند و برای هر تغییر، اصلاحات لازم در $WV(st)$ و $SV(st)$ انجام می‌شود. زمانی که نقطه q

در یک چندضلعی ساده حرکت می‌کند دو نوع رخداد در مورد $SPT(q)$ اتفاق می‌افتد [۱۱]. رخداد اول که به آن رخداد سطح اول گفته می‌شود، زمانی پیش می‌آید که دو فرزند متوالی از ریشه $SPT(q)$ با q هم‌راستا می‌شوند. دراین‌حالت، وابسته به جهت حرکت q یکی از این راس‌ها در درخت $SPT(q)$ فرزند دیگری می‌شود.

نوع دیگر رخدادها زمانی پیش می‌آید که یکی از فرزندان q در درخت $SPT(q)$ و یکی از فرزندان آن راس با q هم‌راستا شوند. در این صورت فرزند دوم را فرزند اصلی می‌نامیم. این نوع رخداد نیز رخداد فرزند اصلی گفته می‌شود. در این حالت، فرزند اصلی در سطح اول درخت و به عنوان فرزند q و پیش یا پس از پدرش قرار می‌گیرد.

بهنگام‌سازی $SV(st)$ به‌ازای هر یک از این رخدادها بدین صورت انجام می‌شود. فرض کنید که یک رخداد سطح اول روی $SPT(t)$ پیش‌آمده است. درنتیجه، دو راس u و v که قبلا از t قابل دید بودند با t هم‌راستا شده‌اند و با ادامه حرکت، راس v دیگر از t قابل دید نیست. چنانچه v قبل از هم‌راستا شدن در $SV(st)$ نباشد، این رخداد باعث تغییر $SV(st)$ نمی‌شود. درغیر اینصورت، v باید از $SV(st)$ حذف شود و راس $V(\overrightarrow{tu}, vv')$ (v' راس بعد از v در $\mathcal{P}$ است) یا $V(\overrightarrow{tu}, \overrightarrow{sv})$ که بین رئوس v و u در $SV(st)$ قرار دارد نیز باید از $SV(st)$ حذف شود.

علاوه‌برآن، براساس نوع راسی که در $SV(st)$ قبل از v قرار دارد باید تغییرات مربوط در $SV(st)$ صورت گیرد: اگر این راس، راسی مانند v'' از نوع A باشد، باید قبل از راس v در $\mathcal{P}$ قرار داشته باشد و دراین‌صورت راس جدید $V(\overrightarrow{tu}, v''v)$ قبل از محل v در $SV(st)$ درج می‌شود. اگر این راس، راسی مانند $V(\overrightarrow{tv}, e)$ از نوع B باشد، از $SV(st)$ حذف و راس $V(\overrightarrow{tu}, e)$ بجای آن در $SV(st)$ درج می‌شود. نهایتا، اگر این راس، راسی مانند $V(\overrightarrow{tv}, \overrightarrow{su'})$ از نوع C باشد از $SV(st)$ حذف و راس جدید $\mathrm{V}(\overrightarrow{tu},\overrightarrow{su'})$ بجای آن درج می‌شود.

حال فرض کنید یک رخداد فرزند اصلی برای $SPT(t)$ پیش‌آمده باشد. در این‌صورت، راس v که قبلا از t قابل دید نبود و راس قابل دید u با t هم‌راستا شده‌اند. اگر v از s قابل دید نباشد (که به معنای این است که v متعلق به $WV(st)$ نیست)، $SV(st)$ تغییر نمی‌کند. درغیراینصورت، v باید در محل مناسب خود در $SV(st)$ درج و دنباله رئوس $SV(st)$ بطور مناسب اصلاح شود. برای این حالت بسادگی قابل اثبات است که در زمان این رخداد u جزء $SV(st)$ است. علاوه‌برآن، یا یک راس نوع B مانند $V(\overrightarrow{tu}, e)$ یا یک راس نوع C مانند $V(\overrightarrow{tu}, \overrightarrow{su'})$ در $SV(st)$ وجود دارد. در هر دو حالت، این راس باید از $SV(st)$ حذف شود و بجای آن در حالت اول راس جدید $V(\overrightarrow{tv}, e)$ و در حالت دوم راس جدید $V(\overrightarrow{tv}, \overrightarrow{su'})$ باید قبل از راس v در $SV(st)$ درج شود. همچنین، برحسب جهت یال vv' (v' راس بعد از v در $\mathcal{P}$ است)، راس جدید $V(\overrightarrow{tu}, \overrightarrow{vv'})$ یا $\mathrm{V}(\overrightarrow{tu},\overrightarrow{sv})$ باید پس از راس v در $SV(st)$ اضافه شود.

تغییرات مربوط به $WV(st)$ نیز به روشی مشابه اعمال می‌شود. فرض کنید که یک رخداد سطح اول روی $SPT(t)$ روی داده است. یعنی دو راس u و v که از t قابل دید هستند با t هم‌راستا

شده‌اند و با ادامه حرکت st، راس v دیگر قابل دید نخواهد بود. اگر v هم‌چنان از st قابل دید باقی بماند کافی است راس $V(\overrightarrow{uv},e)$ به $WV(st)$ اضافه شود. درغیراینصورت، v، $V(\overrightarrow{uv},e)$ و $V(\overrightarrow{tv},e)$ در صورت وجود باید از $WV(st)$ حذف شوند. علاوه‌برآن، اگر $WV(st)$ شامل راس $V(\overrightarrow{tu},e)$ باشد، مقدار آن باید براساس محل تقاطع جدید شعاع $\overrightarrow{tu}$ مجددا محاسبه شود. درغیراینصورت، $V(\overrightarrow{uv},e)$ باید در محل v در دنباله $WV(st)$ درج شود.

برای پردازش یک رخداد فرزند اصلی روی $SPT(t)$، فرض کنید راس v که قبلا از t قابل دید نبود با راس قابل دید u و t هم‌راستا شده است. در این حالت، راس‌های v (درصورتی که در $WV(st)$ وجود نداشته باشد)، $V(\overrightarrow{tu},e)$ و $V(\overrightarrow{uv},e)$ باید در محل درست خود در $WV(st)$ درج شوند. درغیراینصورت، باید راس $V(\overrightarrow{uv},e)$ از $WV(st)$ حذف و به جای آن راس $V(\overrightarrow{tv},e)$ (اگر v راس انعکاسی است) به $WV(st)$ اضافه شود.

۶-۵ تحلیل الگوریتم

بسادگی قابل اثبات است که تغییرات $SV(st)$ و $WV(st)$ به‌ازای هر رخداد در $SPT(s)$ و $SPT(t)$ در زمان $O(\log n)$ انجام‌پذیر است.

لم ۲۲ . هر تغییر دید در $SV(st)$ در زمان $O(\max(\log(|V(s)|),\log(|V(t)|),\log(|SV(st)|)))$ و در نتیجه در زمان $O(\log n)$ انجام‌پذیر است.

اثبات. چنانچه در بخش قبلی گفته شد، برای پردازش هر رخداد، باید روی $V(s)$ و $V(t)$ جستجو کنیم و نیز محل تغییرات را باید در $SV(st)$ پیدا کنیم. سپس، تغییرات در زمان ثابت انجام خواهند شد. بنابراین، هر رخداد در زمان $O(\log(|V(s)|)+\log(|V(t)|)+\log(|SV(st)|))$ روی $SV(st)$ اعمال می‌شود. رخدادهای $SPT(s)$ و $SPT(t)$ نیز به ترتیب در زمان $O(\log(|V(s)|))$ و $O(\log(|V(t)|))$ پردازش و اجرا می‌شوند. با توجه به این‌که $|V(q)|$ و $|SV(st)|$ از مرتبه $O(\log n)$ هستند، هر رخداد مربوط به $SV(st)$ در زمان $O(\log n)$ اجرا می‌شود. □

لم ۲۳ . هر تغییر دید در $WV(st)$ در زمان $O(\log(|WV(st)|))$ و در نتیجه در زمان $O(\log n)$ انجام‌پذیر است.

اثبات. چنانچه در بخش قبلی گفته شد، برای پردازش یک رخداد باید وضعیت یک راس در $V(s)$ و $V(t)$ و نیز محل تغییرات در $WV(st)$ مشخص شود. همچنین باید بررسی کنیم که آیا یک راس از st بصورت ضعیف قابل دید است یا خیر. همه این جستجوها در زمان لگاریتمی قابل انجام است. با داشتن نتیجه این جستجوها، تغییرات مورد نیاز در زمان ثابت انجام می‌شوند. رخدادهای $SPT(s)$ و $SPT(t)$ نیز به ترتیب در زمان $O(\log(|V(s)|))$ و $O(\log(|V(t)|))$ پردازش و

اجرا می‌شوند. با توجه به این‌که $|WV(st)|$ از مرتبه $O(n)$ است، هر رخداد مربوط به $WV(st)$ در زمان $O(\log n)$ اجرا می‌شود. □

نتایج حاصل از این الگوریتم بصورت قضیه زیر قابل بیان است.

قضیه ۱۷ . چندضلعی ساده n راسی $\mathcal{P}$ و پاره‌خط st درون آن‌را می‌توان بگونه‌ای در زمان $O(n \log n)$ پیش‌پردازش کرد که چندضلعی قابل دید قوی و ضعیف st را هنگامی که st در امتداد یک خط حرکت می‌کند در زمان $O(\log n)$ بهنگام‌سازی کنیم. چنانچه جهت حرکت st بتواند تغییر کند، هر تغییر دید در چندضلعی‌های قوی و ضعیف st در زمان $O(\log^2 n)$ اجرا می‌شود.

۶-۶ نتیجه‌گیری

در این فصل الگوریتمی برای نگهداری و بهنگام‌سازی چندضلعی قابل دید ضعیف و قوی یک پاره‌خط واقع در یک چندضلعی ساده بیان شد. در این روش از تغییرات درخت کوتاه‌ترین مسیر دو سر پاره‌خط برای شناسایی و انجام تغییرات دید این چندضلعی‌ها استفاده شد.

اولین قدم در تعمیم این روش، حل این مساله برای چندضلعی‌های بامانع است. سپس، تعمیم مساله برای محیط‌های پویا و نیز حالت سه بعدی قدم‌های بعدی ارزشمندی برای توسعه و ادامه این روش است.

فصل ۷

مجتمع قابلیت دید سه‌بعدی

یکی از داده‌ساختارهای مناسب برای حل مسائل قابلیت دید، مجتمع قابلیت دید است که روابط دیداری بین اشیای یک صحنه را در یک ساختار مناسب نگهداری می‌کند و امکان محاسبه دقیق قابلیت دید یک ناظر را فراهم می‌کند. این روش برای نخستین بار توسط وگتر[۱] و پاچیولا[۲] معرفی شد [۱۰۹]. با استفاده از این روش، چندضلعی قابل دید هر نقطه واقع در درون یک چندضلعی دلخواه را می‌توان با صرف پیش‌پردازش زمانی $O(n \log n + k)$ و حافظه خطی در زمان $O(|V(q)| \log n)$ محاسبه کرد. پارامتر k برابر با اندازه گراف مماس قابلیت دید چندضلعی مرجع است که گراف مماس قابلیت دید در صفحه به گرافی اطلاق می‌شود که در آن رئوس گراف اشیاء هستند و هر مماس مشترک بین دو شی بیانگر یک یال بین رئوس متناظر با آن دو شی است. .

از طرفی، مسائل قابلیت دید در فضای سه‌بعدی دارای پیچیدگی بیشتری نسبت به فضای دو بعدی است و اغلب روش‌های ارائه شده برای آن مبتنی بر روش‌های گرافیکی و حل نادقیق مساله است. با توجه به توانایی مجتمع قابلیت دید در حل مسائل، تلاش‌هایی برای تعمیم مجتمع قابلیت دید دوبعدی به فضای سه‌بعدی صورت گرفته است. مهمترین نتیجه به‌دست آمده در این زمینه در [۴۰] آمده است که مجتمع قابلیت دید سه‌بعدی را برای n شی محدب در زمان $O((n^3 + k) \log n)$ می‌سازد که k اندازه‌ی مجتمع و از مرتبه $O(n^4)$ است. با توجه به پیچیدگی محاسباتی بالای این روش، استفاده از آن در کاربردهای عملی بر اساس نظر خود ارائه کنندگان آن تقریبا ناممکن است. در این فصل ساختار جدیدی برای نگهداری روابط دیداری اشیاء محدب در فضای سه‌بعدی ارائه می‌دهیم که به ترتیب دارای پیچیدگی محاسباتی و حافظه‌ای $O(n^3 \log)$ و $O(n^3)$ است و برای حل مسائل قابلیت دید قابل استفاده است. به عنوان یکی از کاربردهای این ساختار، نحوه محاسبه فضای قابل دید از یک ناظر نقطه‌ای در فضای سه‌بعدی را نیز بیان می‌کنیم.

---

[۱] G. Vegter

[۲] M. Pocchiola

۷-۱ مقدمه

مجتمع قابلیت دید ساختاری توپولوژیک است که روابط دیداری اشیای صحنه را در خود نگاه می‌دارد که پس از ارائه در [۱۰۹] توسط افراد متعددی برای حل مسائل قابلیت دید مورد استفاده قرار گرفت [۱۰۳، ۱۱۵]. مفهوم اصلی در این ساختار، قطعات آزاد بیشینه[۳] است که شامل قطعات با طول بیشینه در فضا است که بجز در انتهای خود با هیچ شی دیگری برخورد نمی‌کنند و به عبارت دیگر دو انتهایشان بر روی دو شی قرار دارند. مجتمع قابلیت دید افراز و دسته‌بندی این قطعات آزاد بیشینه با توجه به اشیائی است که در انتهایشان قرار می‌گیرند.

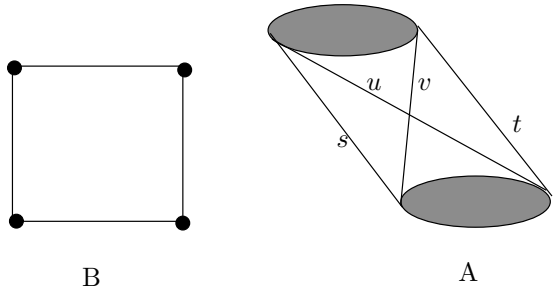
مجتمع قابلیت دید براساس گراف مماس قابلیت دید ساخته می‌شود. این گراف برای مجموعه‌ای از اشیای R بدین ترتیب تعریف می‌شود که اعضای R رئوس این گراف را تشکیل می‌دهند و اگر دو شی دارای مماسی باشند که با هیچ یک از اشیای دیگر تلاقی ندارد، بین رئوس متناظر با آن‌ها یال وجود خواهد داشت. بنابراین، در این گراف بین هر دو گره می‌توانیم ۴ یال داشته باشیم.

مجتمع قابلیت دید در واقع یک نگاشت از محیط موردنظر(چندضلعی) به یک فضای جدید است که دراین نگاشت هر پاره‌خط از محیط مرجع که دو سر آن بر روی مرز دو شی از محیط باشد به یک نقطه از فضای جدید نگاشت می‌شود. علاوه بر آن، در محیط جدید که همان مجتمع قابلیت دید است، یک راس متناظر با یک پاره‌خط مماس بر دو شی از محیط مرجع است و یک یال uv معادل با همه پاره‌خط‌هایی است که یک انتهای آن‌ها از یک طرف (مثلا طرف راست) بر شی r_1 مماس است و انتهای دیگر آن‌ها بر روی مرز یک شی دیگر r_2 قرار دارد. رئوس انتهای یال uv متناظر با دو پاره‌خطی است که از همان طرف(سمت راست) بر شی r_1 مماس هستند و انتهای دیگر آنها یکی از سمت راست و دیگری از سمت چپ بر r_2 مماس است. به این ترتیب، یک وجه از مجتمع قابلیت دید شامل نقاط متناظر با مجموعه تمام پاره‌خط‌هایی است که دو سر آن‌ها بر روی مرز دو شی قرار دارند. در شکل ۷—۱ مجتمع قابلیت دید متناظر با اشیای بخش A در بخش B نشان داده شده است.

برای یکنواختی تعریف فرض می‌شود که همه اشیاء توسط یک شی بزرگ‌تر احاطه شده‌اند و به‌این‌ترتیب، همه فضای بین اشیاء را می‌توان با مجموعه‌ای از پاره‌خط‌ها نشان داد. با این فرض و نیز چنانچه هیچ سه شی متمایزی دارای خط مماس یکسان نباشند، هر راس مجاور با چهار یال و شش وجه و هر یال مجاور با سه وجه و دو راس است.

اندازه داده‌ساختار مجتمع قابلیت دید در صفحه که با k نشان داده می‌شود متناسب با اندازه گراف مماس قابلیت دید اشیاء موجود در صفحه است. چنانچه تعداد اشیای درون صفحه n باشد، الگوریتم کارایی با زمان اجرای $O(n \log n + k)$ برای ساخت آن ارائه شده است [۱۰۹]. به کمک این ساختار چندضلعی قابل دید یک نقطه q واقع در صفحه را می‌توان در زمان $O(|V(q)| \log n)$

[۳] Maximal Free Segments



شکل ۷-۱: مجتمع قابلیت دید دو شی.

محاسبه کرد [۱۰۹].

نکته قابل توجه که باعث کاربرد مجتمع قابلیت دید برای حل مسائل قابلیت دید می‌شود این است که هرگاه در یک مسیر در مجتمع قابلیت دید از یک یال عبور کنیم، چندضلعی قابل دید مربوط به نقطه‌ی متحرکی که در امتداد این مسیر در محیط مبدا در حرکت است نیز در همان لحظه تغییر می‌کند. بنابراین از طریق مجتمع قابلیت دید و تغییرات آن می‌توان چندضلعی قابل دید در محیط‌های پویا را بهنگام کرد. این روش در [۷۶] به‌کار رفته است و براساس آن دو الگوریتم برای تعیین و نگهداری چندضلعی قابل دید یک نقطه متحرک در میان مجموعه‌ای از اشیای ثابت و متحرک ارائه شده است.

تعمیمی از مجتمع قابلیت دید به فضای سه‌بعدی در [۴۱] با عنوان مجتمع قابلیت دید سه‌بعدی معرفی شده است که همانند مجتمع قابلیت دید دو بعدی بر مبنای افراز قطعات آزاد بیشینه است. اندازه‌ی مجتمع قابلیت دید سه‌بعدی که با k نشان داده می‌شود، برای n شی محدب محدود به دو کران $\Omega(n)$ و $O(n^4)$ است و الگوریتمی برای ساخت آن در زمان $O((n^3+k)\log n)$ ارائه شده است [۴۰]. بااین‌حال، به علت بالا بودن پیچیدگی ساختاری مجتمع قابلیت دید سه‌بعدی، استفاده از این ساختار برای یافتن الگوریتم‌های کارا برای حل مسائل قابلیت دید تقریبا ناممکن است [۴۱].

از آنجا که روابط دیداری بین اشیاء در فضای سه‌بعدی به شکل گسترده‌ای پیچیده است، معمولا الگوریتم‌ها و ساختارهای تعریف شده در فضای دوبعدی یا معادلی در فضای سه‌بعدی ندارند (مثلا گراف قابلیت دید) یا مانند مجتمع قابلیت دید تعمیم آن‌ها در این فضا به علت پیچیدگی بالا عملا غیرقابل استفاده‌اند. به عنوان مثال، گراف مماس قابلیت دید مطابق با تعریف آن در صفحه قابل تعمیم به فضای سه‌بعدی نیست و معمولا به گرافی انتزاعی اطلاق می‌شود که در آن رئوس گراف اشیاء هستند و بین رئوس متناظر با دو شی که نسبت به هم قابل دید هستند یک یال وجود دارد [۴۰]. همچنین، مجتمع قابلیت دید سه‌بعدی اگرچه معادل دقیقی برای مجتمع قابلیت دید دوبعدی محسوب می‌شود ولی به علت پیچیدگی محاسباتی بالا کارایی لازم را ندارد. علت تفاوت روابط دیداری در فضای دوبعدی با فضای سه‌بعدی تفاوت ماهیت خط در این دو فضا

است [۴۰]. خط در صفحه فضا را به دو قسمت تفکیک می‌کند که به آن ویژگی تفکیک‌کنندگی[۴] گفته می‌شود ولی در فضای سه‌بعدی این خاصیت برای خط وجود ندارد.

در این فصل روشی جدید برای بیان روابط دیداری در فضای سه‌بعدی ارائه می‌کنیم که آن را می‌توان تعمیمی از مجتمع قابلیت دید دوبعدی در فضای سه‌بعدی دانست که به جای افراز قطعات آزاد بیشینه، صفحات را افراز می‌کند. در این ساختار صفحات بر اساس اشیائی که بر آن‌ها مماس هستند یا آن‌ها را قطع می‌کنند دسته‌بندی می‌شوند. این ساختار دارای اندازه $O(n^3)$ است و در زمان $O(n^3 \log n)$ ساخته می‌شود. این داده‌ساختار را می‌توان برای بیان روابط دیداری بین اشیاء و حل مسائل قابلیت دید به‌کار برد. یک نمونه از کاربرد این ساختار، تعیین $V(q)$ برای یک ناظر نقطه‌ای q است که در زمان $O((|V(q)| + n^2) \log n)$ به‌دست می‌آید.

در ادامه، ابتدا مفاهیم و تعاریف مربوط به مجتمع قابلیت دید در فضای سه‌بعدی را در بخش ۷—۲ بیان می‌کنیم. سپس در بخش ۷—۳، شبه‌گراف قابلیت دید به عنوان یک ساختار پایه‌ای تعریف و الگوریتمی برای ساخت آن پیشنهاد می‌شود. در بخش ۷—۴، نحوه‌ی نگاشت فضای سه‌بعدی هدف به فضای دوگان در ساختار جدید بیان می‌شود. در پایان، ساختار جدید و الگوریتم ساخت آن بیان می‌شود و محاسبه دید به عنوان یکی از کاربردهای آن معرفی و برای آن یک الگوریتم ارائه می‌شود.

۷-۲ قابلیت دید در فضای سه‌بعدی

چنانچه گفته شد، مهم‌ترین تفاوت نقش خط در فضای دوبعدی با نقش آن در فضای سه‌بعدی خاصیت تفکیک‌کنندگی است. در فضای سه‌بعدی صفحه دارای این خاصیت است، یعنی یک صفحه فضا را به دو قسمت مجزا از هم تفکیک می‌کند. بنابراین، سعی می‌کنیم معادلی برای قابلیت دید برحسب صفحات به دست آوریم. مفاهیم پایه قابلیت دید در فضای دوبعدی را می‌توان به شرح زیر خلاصه کرد.

یک پرتو نیم‌خطی جهت‌دار است که از یک نقطه شروع می‌شود. یک قطعه بخشی از یک خط است که میان دو نقطه قرار دارد. به عبارت دیگر یک قطعه در راستای یک خط با دو پرتو در آن راستا و در جهات مختلف مشخص می‌شود. شی قابل دید از یک نقطه و در راستای یک پرتو، اولین شیئی است که توسط این پرتو قطع می‌شود. دو نقطه را متقابلا قابل دید گوییم اگر قطعه‌ی بین آن دو با هیچ شی دیگری برخورد نکند.

حال این مفاهیم را برای فضای سه‌بعدی بر اساس صفحه که طبق خاصیت تفکیک‌کنندگی معادل خط در صفحه است تعریف می‌کنیم. قابلیت دید بین چند نقطه از فضا بر اساس وضعیت قابلیت دید این نقاط در صفحات مشترک آنها تعریف می‌شود. صفحه A در فضای سه‌بعدی و

[۴] Separability

نقطه‌ی p روی آن را در نظر بگیرید. یک پرتو از نقطه‌ی p و در راستای صفحه A را زاویه‌ای تشکیل شده از دو نیم‌خط گذرا از p در A تعریف می‌کنیم. در این حالت یک شی را از یک نقطه‌ی p در راستای صفحه A قابل دید گوییم اگر شی صفحه را قطع کرده باشد. به عبارت دیگر، پرتویی از نقطه p در صفحه وجود داشته باشد که سطح مقطع شی را در بر بگیرد. یک قطعه در یک صفحه را به عنوان یک چندضلعی محدب در آن صفحه در نظر می‌گیریم. قطعه را می‌توان محدود شده توسط تعدادی پرتو (زاویه‌های چندضلعی) در آن صفحه دانست. در ساده‌ترین حالت، یک قطعه مثلثی است که از ترکیب سه پرتو با یال‌های مشترک ایجاد می‌شود. در این حالت قطعه را می‌توان با سه راس مثلث مشخص کرد. همه نقاط یک قطعه متقابلا قابل دید هستند اگر هیچ شیئی آن قطعه را قطع نکرده باشد.

تعاریف بالا پایه‌ای برای تعمیم روابط دیداری دوبعدی به فضای سه‌بعدی خواهند بود.

۷-۳ شبه‌گراف قابلیت دید

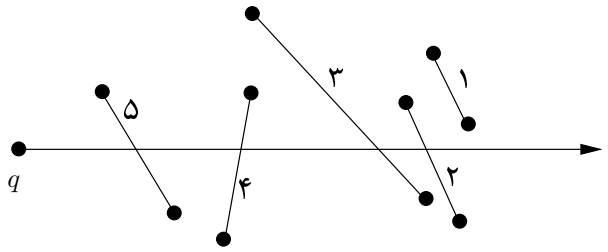
اکنون ساختاری به نام شبه‌گراف تعریف می‌کنیم که آن را می‌توان تعمیم یافته گراف مماس قابلیت دید در فضای سه‌بعدی دانست. برای مجموعه اشیای محدب و هموار $\mathcal{O}$، مجموعه‌ی رئوس این شبه‌گراف، $\mathcal{O}$ است. یال‌های این شبه‌گراف برخلاف گراف عادی که دو راس را به هم مرتبط می‌کند، ارتباط بین سه راس را نشان می‌دهند: به این ترتیب که هر صفحه‌ی مماس بر سه شی O_1، O_2 و O_3 یک یال $\{O_1, O_2, O_3\}$ را مشخص می‌کند هرگاه سه نقطه‌ی مماس متقابلا قابل دید باشند (هیچ شیئی قطعه‌ی تشکیل شده توسط این سه نقطه را قطع نکند). با توجه به وضعیت اشیاء، هر سه راس این شبه‌گراف می‌توانند حداکثر ۸ یال مشترک داشته باشند.

اندازه شبه‌گراف مربوط به n شی که با k نشان داده می‌شود، متناظر با تعداد یال‌های شبه‌گراف است که $O(n^3)$ و $\Omega(n)$ است. در ادامه الگوریتمی با مرتبه‌ی زمانی $O(n^3 \log n)$ برای ساخت این شبه‌گراف ارائه می‌دهیم. با توجه به تناظری که بین گراف قابلیت دید دوبعدی و شبه‌گراف قابلیت دید وجود دارد، وجود الگوریتم‌هایی با مرتبه‌ی $O(k \log n)$ و $O(k + n \log n)$ نیز برای آن قابل تصور است.

الگوریتمی که برای ساخت شبه‌گراف قابلیت دید ارائه می‌دهیم متناظر است با الگوریتم لی [۵] برای ساخت گراف قابلیت دید دوبعدی که آن را در زیر بصورت خلاصه بیان می‌کنیم [۸۹].

فرض کنید که مجموعه‌ای از پاره‌خط‌ها در صفحه مفروض است. برای هر راس بقیه‌ی رئوس را بر اساس زاویه آن‌ها حول این راس مرتب می‌کنیم. سپس آن‌ها را به ترتیب بررسی می‌کنیم و همزمان لیستی از پاره‌خط‌هایی که در آن لحظه می‌بینیم نگهداری می‌کنیم. چنانچه راس جدیدی که به آن برخورد می‌کنیم متعلق به پاره‌خط اول لیست باشد آن را گزارش می‌کنیم وگرنه

[۵] D.T. Lee



شکل ۷-۲: تعیین گراف مماس قابلیت دید براساس الگوریتم لی.

می‌دانیم که این راس توسط پاره‌خطی پوشانده شده است. شکل ۷—۲ ایده‌ی کلی الگوریتم را نشان می‌دهد. در این‌جا لیست پاره‌خط‌ها $\{۵, ۴, ۳, ۲\}$ است.

حال الگوریتم ساخت شبه‌گراف را بیان می‌کنیم. برای سادگی اشیای موجود در فضا را کروی در نظر می‌گیریم. به ازای هر دو شی $O_۱$ و $O_۲$ از فضا الگوریتم زیر را برای تشخیص شبه‌یال‌هایی که با دو راس متناظر با این دو شی تشکیل می‌شوند اجرا می‌کنیم.

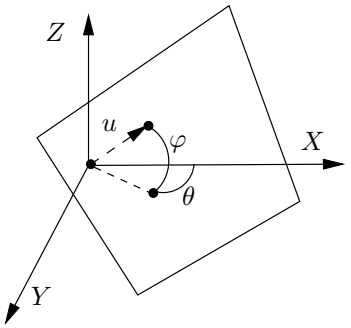
یک صفحه مماس بر $O_۱$ و $O_۲$ را به عنوان مرجع در نظر می‌گیریم و کلیه صفحات مماس بر $O_۱$ و $O_۲$ و یک شی دیگر از فضا را براساس زاویه آنها با صفحه مرجع مرتب می‌کنیم. سپس رئوس فضا را براساس ترتیبی که به دست می‌آید جاروب می‌کنیم که این کار معادل روبش فضا توسط صفحه‌ای دورانی و مماس بر $O_۱$ و $O_۲$ است. در این‌جا هم لیستی از اشیائی که توسط صفحه‌ی روبش قطع شده‌اند نگهداری می‌کنیم و با برخورد با هر شی جدید، موقعیت آن را با بقیه‌ی مثلث‌های درون لیست بررسی می‌کنیم و بر حسب مورد آن را به لیست اضافه یا از لیست خارج می‌کنیم.

زمان لازم برای مرتب کردن صفحات مماس $O(n \log n)$ و زمان لازم برای هر یک از عملیات درج یا حذف از لیست، $O(\log n)$ است. در نتیجه الگوریتم روبش در زمان $O(n \log n)$ انجام می‌شود.

با توجه به این که $n^۲$ زوج شی وجود دارد، $O(n^۲)$ بار روبش انجام می‌شود و زمان اجرای کل الگوریتم از مرتبه‌ی $O(n^۳ \log n)$ است.

۷-۴ نگاشت صفحه در فضای دوگان

هر صفحه در فضا را می‌توان با سه پارامتر مشخص کرد. در فضای دوگان این سه پارامتر مشخص کننده یک نقطه هستند. بنابراین، هر صفحه در فضای مرجع به یک نقطه در فضای دوگان نگاشت می‌شود. پارامترهایی که ما برای مشخص کردن یک صفحه تعریف می‌کنیم شامل طول خط عمود بر صفحه و گذرنده از مبدا (u)، زاویه این خط با صفحه XY (φ) و زاویه تصویر این خط در صفحه



شکل ۷–۳: پارامتری کردن صفحه با استفاده از دو زاویه برای جهت خط عمود بر صفحه و فاصله‌ی صفحه تا مبدا.

XY با محور X (θ) است. این پارامترها در شکل ۷—۳ نشان داده شده‌اند. برای حالتی که صفحه از مبدا می‌گذرد خط گذرا از مبدا عمود بر صفحه را استفاده می‌کنیم.

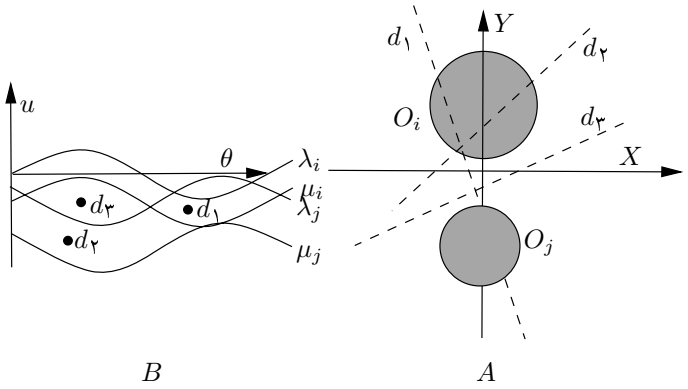
اکنون این صفحه را به نقطه‌ی (θ, φ, u) در فضای دوگان نگاشت می‌کنیم. این نگاشت فضای صفحات را به $[0, 2\pi] \times [-\pi/2, \pi/2] \times R$ نگاشت می‌کند.

۷–۵ مجتمع قابلیت دید جزئی

با توجه به پیچیدگی محاسباتی بالای مجتمع قابلیت دید سه بعدی، در این بخش براساس تعاریف قابلیت دید در فضای سه بعدی ساختاری ارائه می‌دهیم که آن را می‌توان گونه خاصی از تعمیم مجتمع قابلیت دید دوبعدی به فضای سه‌بعدی دانست. این ساختار که به آن مجتمع قابلیت دید جزئی سه‌بعدی می‌گوییم ابتدا در حالت‌های خاص معرفی می‌کنیم و سپس در حالت کلی آن را تعریف می‌کنیم و الگوریتمی برای ساختن آن ارائه می‌دهیم.

یک کره و یک صفحه در فضا را در نظر بگیرید و فرض کنید که صفحه در فضا به طور پیوسته قابل حرکت است. در این حرکت مرز حالاتی که صفحه با یک شی برخورد می‌کند با حالاتی که با آن برخورد ندارد مواقعی است که صفحه بر یک شی مماس می‌شود. بر این اساس می‌توان این مرز را در فضای دوگان محاسبه کرد و صفحات فضا را با این مرز تقسیم‌بندی کرد.

صفحه‌ای که بر یک کره مماس است برای این‌که در ضمن حرکت بر آن کره مماس بماند، دو درجه آزادی برای حرکت دارد. در ادامه منظور ما از جهت صفحه جهت خط عمود بر صفحه است که با دو پارامتر (θ, φ) مشخص می‌شود. برای هر جهت (θ, φ) در فضای سه‌بعدی دو صفحه $(\theta, \varphi, \lambda(\theta, \varphi))$ و $(\theta, \varphi, \mu(\theta, \varphi))$ مماس بر این کره وجود دارد. $\lambda(\theta, \varphi)$ و $\mu(\theta, \varphi)$ دو رویه را در فضای دوگان مشخص می‌کنند. این دو رویه مرزهایی را در فضای دوگان به وجود می‌آورند که صفحات فضا را بر اساس اشیائی که با آن‌ها برخورد می‌کنند تقسیم می‌کنند. به عنوان مثال، هر



شکل ۷–۴: برشی از فضای مرجع و فضای دوگان آن.

صفحه (θ, φ, u) که رابطه‌ی $\mu(\theta,\varphi) < u < \lambda(\theta,\varphi)$ برای آن برقرار باشد با کره برخورد می‌کند. این رویه‌ها که از نقاط دوگان صفحات مماس بر شی تشکیل شده‌اند، رویه مماسی و حجم محصور بین رویه‌های یک شی را حجم مماسی آن شی می‌گوییم.

شکل ۷—۴ برشی از فضا و نگاشت آن در فضای دوگان را نشان می‌دهد. فضای اصلی (قسمت A) از دو شی O_i و O_j تشکیل شده است که در این شکل تقاطع سه صفحه D_1، D_2 و D_3 با صفحه xy نشان داده شده است. در قسمت B، φ–برشی (برشی از فضای دوگان که دارای φ ثابت است) از صفحات مماس بر دو شی دیده می‌شود. در اینجا فرض کرده‌ایم که نقاط متناظر با صفحات D_i، $i = ۱, ۲, ۳$، در فضای دوگان روی برش ما از این فضا قرار دارند. چنانچه می‌بینیم صفحه D_1 که هر دو شی را قطع کرده است درون حجم‌های مماسی هر دو شی قرار دارد و صفحه D_3 که اشتراکی با دو شی ندارد خارج حجم‌های مماسی آن‌هاست. لازم به ذکر است که دامنه‌ی θ، $[۰, ۲\pi]$ و دامنه φ، $[-\pi/۲, \pi/۲]$ است.

برای حالت فوق، نقاط اشتراک رویه‌های مماسی دو شی یک منحنی تک بعدی ایجاد می‌کند که متناظر صفحاتی است که بر هر دو شی مماس هستند. چنانچه فضا شامل سه شی باشد تقاطع رویه‌های مماسی دارای حداکثر ۸ نقطه است که هر نقطه متناظر با صفحه مماس بر سه شی است.

با همین استدلال، برای هر چینش دلخواه از اشیاء، رویه‌های مماسی اشیای مختلف، فضای دوگان را به بخش‌های پیوسته‌ای تقسیم می‌کنند که تعلق یک صفحه به هر بخش نمایانگر اشیائی است که صفحه با آن‌ها برخورد می‌کند.

تعریف کامل مجتمع قابلیت دید جزئی را براساس نگاشت بالا تعریف می‌کنیم. در فضای سه‌بعدی n شی محدب را در نظر بگیرید. برای یکپارچگی ساختار، شی نامحدودی را که اشیاء صحنه را در بر گرفته است به فضا اضافه می‌کنیم. مجتمع قابلیت دید را تقسیم‌بندی مجموعه قطعات آزاد بیشینه براساس اشیائی که رئوس این قطعات بر آن‌ها واقع شده‌اند تعریف می‌کنیم که منظور از قطعه آزاد بیشینه قطعات مثلث شکلی هستند که به‌جز در رئوس خود با شی دیگری

برخورد نمی‌کنند. مجتمع قابلیت دید جزئی مانند مجتمع قابلیت دید دوبعدی از چند نوع عنصر تشکیل شده است. این عناصر ۰ تا ۴ بعدی هستند که به ترتیب افزایش بعد عبارتند از راس (جزء صفربعدی که متناظر است با یک صفحه مماس بر سه شی)، ضلع (جزء یک‌بعدی که متناظر با مجموعه‌ی صفحاتی است که بر دو شی مماس و با شی سومی برخورد می‌کنند)، وجه (جزء دوبعدی که متناظر مجموعه صفحاتی است که بر یک شی مماس و با دو شی دیگر برخورد می‌کنند) و حجم (جزء سه‌بعدی که متناظر مجموعه صفحاتی است که به سه شی برخورد می‌کنند).

در فضای دوگان هر یک از اجزای مجتمع مجاور و یا محدود شده توسط تعداد محدودی از اجزای دیگر آن است. به عنوان مثال، روابط زیر در مورد اجزای مجتمع جزئی برقرارند: هر راس مجاور ۶ یال و ۱۲ وجه است؛ هر یال به دو راس محدود شده است؛ هر وجه با ۴ راس محدود شده و مجاور ۴ یال است؛ و هر حجم با دو زنجیره وجه، یال و راس محدود شده است.

برای جمله‌ی اول، سه شی A، B و C و مماس مشترک آن‌ها را در نظر بگیرید. تعداد یال‌های مجاور این راس در فضای دوگان (مماس مشترک) این گونه به دست می‌آید که اگر صفحه‌ی مماس را بر دو شی A و B مماس نگاه داریم و آن را حرکت دهیم، دو یال به دست می‌آید: یک یال مجموعه‌ی صفحات مماس بر A و B که با C برخورد می‌کنند و یال دیگر مجموعه‌ی صفحات مماس بر A و B که با C برخورد نمی‌کنند. به این ترتیب ۶ یال مجاور این راس به دست می‌آید. در مورد جمله‌ی سوم چون هر وجه مجموعه‌ی قطعاتی است که بر یک شی مماس و با دو شی دیگر برخورد می‌کند، برای هر یک از آن دو شی صفحات این مجموعه در دو وضعیت بر این شی مماس می‌شوند (بالا و پایین شی) و هر یک از این حالات نشان‌دهنده‌ی یک یال هستند.

۷-۵-۱ پیچیدگی محاسباتی مجتمع قابلیت دید جزئی

همان‌طور که گفته شد، تعداد اجزایی از مجتمع که مجاور یک جزء دیگر هستند محدود به یک عدد ثابت است. بنابراین، اندازه مجتمع با تعداد رئوس آن مشخص می‌شود. بسادگی قابل بررسی است که تعداد رئوس مجتمع برابر است با $\Omega(n)$ و $O(n^۳)$ که n تعداد اشیاء محدب موجود در فضا است. در حقیقت اندازه‌ی مجتمع قابلیت دید جزئی متناظر با اندازه شبه‌گراف قابلیت دید آن فضا است.

عناصر مجتمع از راس، یال، وجه و حجم تشکیل شده‌اند. هر راس دارای ۳ اشاره‌گر به اشیائی است که بر آن‌ها مماس است. همچنین دارای ۶ اشاره‌گر به یال‌های مجاور و ۱۲ اشاره‌گر به حجم‌های مجاور خود در فضای دوگان است. به همین ترتیب هر یال دو اشاره‌گر به اشیائی که بر آن‌ها مماس است، یک اشاره‌گر به شیئی که با آن برخورد می‌کند و اشاره‌گرهایی به اجزای مجاور خود در فضای دوگان دارد. ساختار وجه و حجم هم به همین ترتیب به دست می‌آیند.

ساختن مجتمع قابلیت دید را می‌توان با یک الگوریتم پویش رئوس آن انجام داد. وقتی که یک راس پویش می‌شود، رابطه‌ی بین یال‌ها و وجه‌ها و حجم‌ها باید بهنگام شود. الگوریتمی را که برای ساخت شبه‌گراف قابلیت دید بیان شد را می‌توان در این‌جا به کار برد و راس‌های مجتمع را

پویش کرد. با توجه به پیچیدگی زمانی این الگوریتم یعنی $O(n^3 \log n)$ و محدود بودن رابطه‌ی هر جزء مجتمع با اجزای دیگر، ساخت مجتمع قابلیت دید را نیز می‌توان در زمان $O(n^3 \log n)$ انجام داد.

۷-۵-۲ کاربرد مجتمع قابلیت دید جزئی در محاسبه‌ی فضای قابل دید

تقریبا همه الگوریتم‌هایی که برای حل مساله تعیین فضای قابل دید از یک ناظر در فضای سه بعدی ارائه شده‌اند مربوط به کاربردهای گرافیکی هستند که براساس موارد کاربرد خاص طراحی شده‌اند و دقت آنها در حد دقت مورد نیاز در صفحه‌ی تصویر یا اشیاء قابل دید است. در [۴۰] این الگوریتم‌ها مرور شده‌اند. همچنین در [۴۱] نحوه‌ی نمایش دید یک ناظر نقطه‌ای براساس مجتمع قابلیت دید سه بعدی بیان شده است که به علت پیچیدگی بالای آن الگوریتم قابل استفاده‌ای در کاربرد برای آن ارائه نشده‌است.

در این بخش، با استفاده از مجتمع قابلیت دید جزئی الگوریتمی برای مساله محاسبه فضای قابل دید یک ناظر نقطه‌ای بیان می‌کنیم. در ادامه، برای سهولت به مجتمع قابلیت دید جزئی با نام مجتمع اشاره می‌کنیم. محاسبه‌ی فضای قابل دید یک ناظر نقطه‌ای، به معنای مشخص کردن اشیائی است که پرتوهای فرستاده شده از نقطه‌ی دید آنها را می‌بینند. برای این کار باید دسته پرتوهایی که اشیاء یکسانی را می‌بینند از هم تفکیک شوند و به بیان دیگر در یک روبش، مرز تغییر این اشیاء را مشخص کرد.

فرض می‌کنیم نقطه‌ی دید خارج پوش محدب[۶] اشیاء باشد. صفحه‌ای گذرا از نقطه‌ی دید را که با هیچ یک از اشیاء برخورد نکند به عنوان صفحه روبش در نظر می‌گیریم. اگر این صفحه را حول محوری گذرا از نقطه‌ی دید دوران دهیم، صفحات متناظر آن در فضای دوگان یک منحنی به نام p_v را تشکیل می‌دهند. این منحنی وجه‌های مجتمع را در نقاطی قطع می‌کند. برخورد با مجتمع در یک وجه متناظر صفحه‌ای مماس بر یک شی است. برخورد در یک یال (اشتراک دو وجه) متناظر صفحه‌ای مماس بر دو شی و برخورد با یک راس (اشتراک سه وجه) متناظر صفحه‌ای مماس بر سه شی است.

در حین حرکت صفحه‌ی روبش، سطح مقطع اشیائی که با آن برخورد می‌کنند وضعیت دوبعدی پویایی از اشیاء بر روی آن ایجاد می‌کنند. مجتمع قابلیت دید دوبعدی این صحنه‌ی پویا را در لحظه نگهداری می‌کنیم. با داشتن مجتمع قابلیت دید دوبعدی سطح مقطع این اشیاء می‌توان منظره‌ای را که ناظر نقطه‌ای در آن صفحه می‌بیند مشخص کرد. از طرفی، در حین حرکت چنانچه موقعیت توپولوژیک این سطح مقطع‌ها نسبت به هم تغییر نکند مجتمع قابلیت دید دوبعدی هم تغییر نخواهد کرد و تغییری در اشیاء قابل دید از ناظر ایجاد نمی‌شود. از این مطلب نتیجه می‌شود که برای محاسبه فضای قابل دید کافی است که مجتمع قابلیت دید دوبعدی صحنه‌ی روی صفحه‌ی

[۶] Convex Hull

روبش را در هنگام روبش نگهداری کنیم و هرگاه تغییری در این مجتمع بوجود آمد، فضای قابل دید را هم در آن صفحه بهنگام کنیم.

مادام که صفحه روبش با وجهی از مجتمع برخورد نداشته باشد تعداد اشیاء برخورد کرده با صفحه روبش و در نتیجه تعداد سطح مقطع اشیاء در صفحه روبش ثابت است و نگهداری دید در حین حرکت صفحه روبش معادل نگهداری دید یک نقطه در صفحه‌ای شامل اشیاء متحرک است. این کار را در هر رخداد تغییر دید می‌توان در $O(\log m)$ انجام داد که m تعداد اشیاء برخورد کرده با صفحه روبش است [۱۱۵] .

با ادامه دوران صفحه روبش، منحنی p_v در نقطه‌ای با مجتمع برخورد می‌کند. اگر برخورد p_v با i وجه مجتمع باشد ($i = ۱, ۲, ۳$، به ترتیب متناظر وجه، یال و نقطه) به این معنی است که k شی شروع به برخورد با صفحه روبش کرده‌اند و l شی دیگر برخورد خود را خاتمه داده‌اند ($k + l = i$). شروع برخورد با شی به این معنی است که یک شی را باید در مجتمع قابلیت دید دوبعدی وارد کنیم. این کار را می‌توان در $O(v \log n)$ انجام داد که v اندازه دید است [۱۱۵]. هنگامی که یک شی برخورد خود را با صفحه روبش خاتمه می‌دهد وجه‌های متناظر آن در مجتمع قابلیت دید دوبعدی از بین می‌روند. این حذف را می‌توان در $O(v)$ انجام داد که باز هم v اندازه دید در مجتمع دوبعدی است [۱۱۵].

ابتدا هزینه‌ی تغییرات در نقاط برخورد p_v با مجتمع را به دست می‌آوریم. تعداد این رخدادها $O(n)$ است که آن‌ها را می‌توان در یک صف با هزینه‌ی $O(n \log n)$ نگهداری کرد. هر شی یک بار شروع به برخورد با صفحه روبش می‌کند و یک بار برخورد خود را قطع می‌کند و اندازه دید در صفحه روبش از مرتبه تعداد اشیاء برخورد کرده با صفحه روبش در آن لحظه است. بنابراین، هزینه زمانی انجام همه این رخدادها $O(n^۲ \log n)$ است.

حال هزینه‌ی نگهداری مجتمع قابلیت دید دوبعدی و دید در آن را در مدتی که تعداد اشیاء برخورد کرده با صفحه روبش ثابت است محاسبه می‌کنیم. فرض کنید در طول قطعه‌ای از p_v که با وجهی از مجتمع برخورد نمی‌کند، تعداد اشیاء برخورد کرده با صفحه روبش m باشد. مجتمع قابلیت دید سطح مقطع اشیاء در صفحه روبش و یا چندضلعی قابل دید ناظر نقطه‌ای زمانی تغییر می‌کند که یا یک دومماسی (خطی که بر دو شی مماس است) بر شی سومی هم مماس شود که به معنی تغییر در مجتمع قابلیت دید دوبعدی است، و یا مماس بر یک شی از نقطه دید، بر شی دیگری هم مماس شود که به معنی تغییر در دید است. تعداد دو مماسی‌ها در صفحه روبش $O(m^۲)$ و تعداد مماس‌ها از نقطه دید $O(m)$ است. در طول مسیر p_v سطح مقطع هر شی در مدتی که آن شی با صفحه روبش برخورد دارد، $O(n)$ دوممماسی با سایر سطوح مقطع روی صفحه روبش می‌سازد. با توجه به این‌که اشیاء محدب فرض شده‌اند و با هم برخوردی ندارند، در طول حرکت صفحه روبش این دوممماسی‌ها می‌توانند $O(n)$ بار توسط اشیاء دیگر قطع شوند. تعداد این رخدادها برای تمام اشیاء $O(n^۲)$ است و در این رخدادها ساختار توپولوژیک مجتمع قابلیت دید دوبعدی تغییر می‌کند و باید مجتمع قابلیت دید را بهنگام کنیم. هزینه این بهنگام‌سازی $O(\log m)$ است [۱۱۵] و در نتیجه، هزینه کل آن‌ها $O(n^۲ \log n)$ خواهد بود. نوع دیگر رخداد تغییر دید، همان‌طور که گفتیم، وقتی

است که خط مماس از نقطه دید بر یک شی توسط شی دیگری قطع شود. دراین حالت وضعیت دید شی قطع کننده نسبت به نقطه دید تغییر خواهد کرد. چون در هر یک از این رخدادها دید تغییر می‌کند پس تعداد کل این رخدادها از مرتبه اندازه چندضلعی قابل دید خواهد بود. بهنگام‌سازی مجتمع قابلیت دید دوبعدی در این نوع رخدادها هم $O(\log m)$ است [۱۱۵].

از بررسی‌های بالا نتیجه می‌شود که اگر اندازه چندضلعی قابل دید ناظر q را با $|V(q)|$ نمایش دهیم، هزینه حساس به خروجی الگوریتم ارائه شده $O((|V(q)|+n^2)\log n)$ است. مشخص است که مقدار پردازش برای محاسبه اشیای قابل دید در این روش اگر چه نسبت به الگوریتم‌های قبلی بهتر است، ولی در فضای سه‌بعدی هنوز بالاست و برای استفاده در کاربرد نیاز به بهبود دارد.

۷-۶ نتیجه‌گیری

در این فصل داده‌ساختار مجتمع قابلیت دید را به عنوان یک ابزار قدرتمند برای توصیف روابط دیداری اشیای موجود در فضا بررسی کردیم. مجتمع قابلیت دید که نخستین بار در [۱۰۹] معرفی شده است، روابط دیداری اشیاء موجود در یک صفحه را به صورت کامل توصیف می‌کند. این ساختار مبتنی بر قطعات آزاد بیشینه‌ای است که همه نقاط روی آنها دارای دید یکسانی در آن امتداد هستند. در [۴۱] این داده‌ساختار برای استفاده در فضای سه‌بعدی تعمیم داده شده است. ساختاری که به این ترتیب ایجاد می‌شود دارای پیچیدگی بسیاری است و عملا غیر قابل استفاده است. این امر مربوط به تفاوت بین ماهیت دید در فضای سه‌بعدی نسبت به فضای دوبعدی است.

به همین دلیل تعریف جدیدی از قابلیت دید فضای سه‌بعدی ارائه شد که ضمن اینکه تعاریف موجود در فضای دوبعدی در آن حفظ شده است، از پیچیدگی محاسباتی آن نیز کاسته شده است تا قابل استفاده در کاربردهای عملی باشد. این ساختار بر خلاف معادل دوبعدی آن، براساس صفحه‌های مماسی و قابلیت دید براساس این صفحات است. اندازه‌ی این ساختار برای m شی محدب در فضای سه‌بعدی $O(n^3)$ است و در زمان $O(n^3\log n)$ قابل ساخت است و الگوریتمی برای ساخت آن نیز ارائه شد.

ساختار پیشنهاد شده می‌تواند برای حل مسائل قابلیت دید استفاده شود که به عنوان یک نمونه از کاربردهای آن، فضای قابل دید از یک ناظر نقطه‌ای محاسبه شده است. این کار برای ناظر q با فضای قابل دید $V(q)$ در زمان $O((|V(q)|+n^2)\log n)$ انجام می‌شود.

نتایج بیان شده در این فصل از دو جهت قابل توسعه هستند: نخست یافتن موارد کاربرد ساختار مجتمع قابلیت دید ارائه شده و حل مسائل کلاسیک این حوزه به کمک آن است که یک نمونه از آن در بخش قبلی بیان شد. دیگر، بررسی و بهبود خود ساختار است که ممکن است منجر به کاهش پیچیدگی محاسباتی آن یا ایجاد سهولت در بکارگیری آن در کاربردهای مختلف شود.

فصل ۸

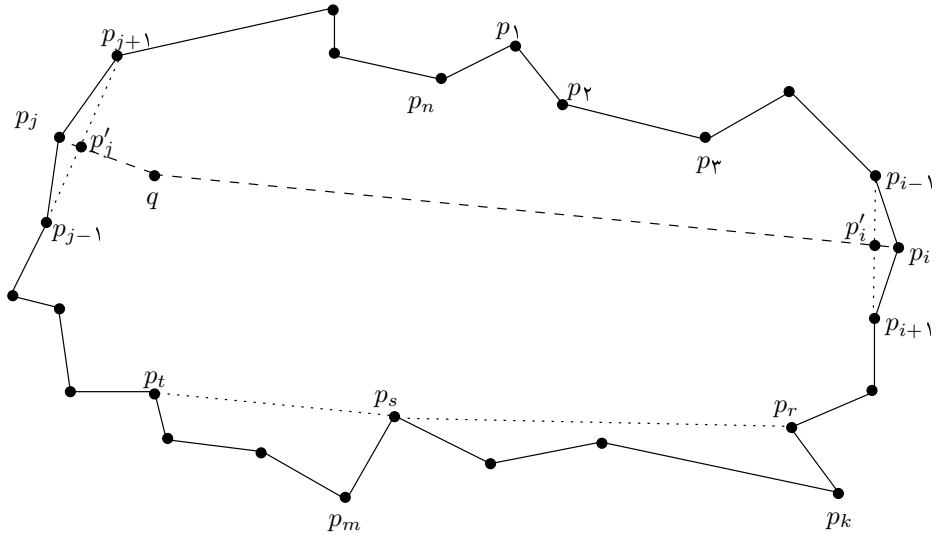
ساده‌سازی چندضلعی قابل دید

مرز بیرونی ناحیه قابل دید ممکن است از رئوس و پاره‌خط‌های زیادی تشکیل شده باشد. این ناحیه‌ها معمولا بر روی صفحاتی نمایش داده می‌شوند که دقت آنها محدود است و در نتیجه، هنگام نمایش شکل تقریبی این مرز نشان داده می‌شود. علاوه‌برآن، نگهداری و انتقال حجم زیادی از این اطلاعات در برخی کاربردها عملی نیست و تهیه تقریب ساده شده‌ای از مرز $V(q)$ برای کاربردهای واقعی کافی است. این مساله حالت خاصی از مساله معروف ساده‌سازی مسیر است که معیارها و الگوریتم‌های متعددی برای آن ارائه شده‌است. بااین‌وجود، هیچ یک از این الگوریتم‌ها برای ساده‌سازی مرز دید یک ناظر که موقعیت ناظر در آن اهمیت کلیدی دارد، مناسب نیستند.

در این فصل، ابتدا معیاری برای ساده‌سازی مرز دید ارائه می‌شود که وابسته به موقعیت ناظر عمل می‌کند و بخش‌هایی از مرز که به ناظر نزدیکترند مهم‌تر از بخش‌های دور از ناظر در نظر گرفته می‌شوند. برای محاسبه این معیار در مرز یک ناحیه الگوریتمی کارا ارائه می‌شود که بدون افزایش زمان و حافظه، در الگوریتم‌های موجود ساده‌سازی قابل استفاده است. این معیارها و الگوریتم‌ها برای ساده‌سازی چندضلعی قابل دید ناظر نقطه‌ای ارائه شده‌اند و سپس برای ساده‌سازی چندضلعی قابل دید ضعیف و قوی ناظر پاره‌خطی تعمیم داده شده‌اند.

۸-۱ مقدمه

در کاربردهای واقعی، قدرت دید ناظر معمولا محدود است و با دور شدن اشیاء میزان وضوح آنها نیز کمتر می‌شود. بنابراین، جزئیات موجود در جاهایی از مرز $V(q)$ که از q دورتر هستند، نسبت به این جزئیات در فاصله نزدیک به q، از اهمیت کمتری برخوردار هستند. از طرف دیگر، فضای لازم برای نگهداری این جزئیات و زمان لازم برای پردازش آنها ممکن است بیشتر از مقدار حافظه و زمان موجود باشد. علاوه‌برآن، دقت صفحات نمایش واقعی محدود است و امکان نمایش جزئیات



شکل ۸–۱: در ساده‌سازی $V(q)$، راس p_i که دورتر از q قرار دارد، انتخاب مناسب‌تری نسبت به راس p_j است که در نزدیکی q قرار دارد.

موجود در مرز ناحیه دید را ندارد و عملا تقریبی از آن نشان داده می‌شود.

به همین دلیل، تقریب زدن مرز ناحیه قابل دید با یک چندضلعی ساده‌تر روشی مناسب برای افزایش کارایی است. مرز این ناحیه از تعدادی پاره‌خط متصل به هم تشکیل شده‌است که فاصله آنها نسبت به ناظر متفاوت است. در نتیجه چندین پاره‌خط که در فاصله دورتری از ناظر قرار دارند قابل جایگزینی با یک پاره‌خط هستند. به عنوان مثال، $V(q)$ در شکل ۸–۱ از پاره‌خط‌های $\langle p_1p_2, p_2p_3, \ldots, p_np_1\rangle$ تشکیل شده‌است و پاره‌خط‌های $p_{i-1}p_i$ و p_ip_{i+1} بسیار دورتر از پاره‌خط‌های $p_{j-1}p_j$ و p_jp_{j+1} نسبت به ناظر قرار دارند. بنابراین، q می‌تواند محل دقیق راس p_j را تشخیص دهد در حالیکه p_i ممکن است به عنوان یک نقطه از پاره‌خط $p_{i-1}p_{i+1}$ به نظر برسد. بنابراین، اگر بخواهیم $V(q)$ را با حذف یک راس از آن، به صورت ساده‌تر نگهداری کنیم p_i انتخاب مناسب‌تری نسبت به p_j برای حذف شدن است. در این فصل، این نوع ساده‌سازی را برای مرز ناحیه قابل دید از یک ناظر نقطه‌ای و پاره‌خطی بررسی می‌کنیم.

این مساله حالت خاصی از مساله معروف ساده‌سازی مسیر است که الگوریتم‌های متعددی برای آن ارائه شده‌است. در این الگوریتم‌ها، مسیر اولیه با یک مسیر دیگر که تعداد راس‌های آن کمتر است جایگزین می‌شود. خطای این ساده‌سازی بر حسب میزان شباهت این دو چندضلعی براساس معیارهای مختلفی محاسبه می‌شود.

در مسائل ساده‌سازی دو هدف دنبال می‌شود که بعنوان دو مساله ساده‌سازی با کمترین راس و ساده‌سازی با کمترین خطا مطرح می‌شوند. در مساله ساده‌سازی با کمترین راس، حداکثر خطای مجاز δ داده شده است و می‌خواهیم ساده‌سازی را با کمترین تعداد راس بگونه‌ای انجام دهیم که

خطای آن از مقدار δ بیشتر نشود. در مساله ساده‌سازی با کمترین خطا، به ازای مقدار داده‌شده k، می‌خواهیم ساده‌سازیی با حداکثر k راس پیدا کنیم که کمترین خطا را داشته باشد. مساله ساده‌سازی با کمترین خطا را به صورت خلاصه‌تر مساله کمترین خطا و مساله ساده‌سازی با کمترین راس را به صورت خلاصه‌تر مساله کمترین راس می‌نامیم. مساله کمترین خطا را می‌توان با جستجوی دودویی روی نتایج مساله کمترین راس حل کرد. بنابراین، در اینجا فقط مساله کمترین راس را بررسی می‌کنیم.

دو دسته اصلی از مساله ساده‌سازی تعریف شده‌اند: ساده‌سازی مقید و بی‌قید. در ساده‌سازی مقید، رئوس مربوط به مسیر ساده‌شده حتما از بین رئوس مسیر اصلی انتخاب می‌شوند ولی در ساده‌سازی بی‌قید، این محدودیت وجود ندارد. برخی از نتایج مربوط به ساده‌سازی بی‌قید در منابع [۵۸، ۶۳، ۶۶، ۷۹] ارائه شده‌است. برای حالت مقید مساله، الگوریتم‌های بیشتری ارائه شده است[۷، ۹، ۲۸، ۳۷، ۵۷، ۷۳، ۸۰، ۹۶]. با توجه به اینکه زمان اجرای اغلب این الگوریتم‌ها از مرتبه $O(n^۲)$ و بالاتر است، تلاش‌هایی برای حل تقریبی حالت مقید این مساله نیز انجام شده است [۷].

در این الگوریتم‌ها، از معیارهای مختلفی برای تعیین مقدار خطای ساده‌سازی استفاده شده است. فاصله هاوس‌دورف برای $L_۱, L_۲, L_\infty$ و مساحت جابجاشده از مهمترین معیارهای تعیین خطای ساده‌سازی است. همچنین، معیارخطای فِرشه نیز در برخی الگوریتم‌ها استفاده شده است [۵۷، ۹]. این الگوریتم‌ها و معیارهای خطا در بخش ۹—۱ بطور خلاصه معرفی می‌شوند.

در مورد مرز ناحیه قابل دید، موقعیت ناظر عامل مهمی در تعیین رئوس قابل حذف هنگام ساده‌سازی است. این در حالی است که در هیچ یک از معیارهای ساده‌سازی موجود، موقعیت ناظر در نظر گرفته نمی‌شود. در نتیجه، لازم است که معیار خطای جدیدی ارائه شود که فاصله ناظر تا نقاط روی مرز ناحیه قابل دید را در تعیین خطای ناشی از حذف یک راس در نظر بگیرد.

بدین منظور، در این فصل ابتدا یک معیار خطا ارائه می‌شود که مبتنی بر این فاصله است. نشان می‌دهیم که چگونه می‌توان این مقدار خطا را بصورت کارایی محاسبه کرد و بدون افزایش هزینه زمانی یا حافظه‌ای، آن‌را در الگوریتم‌های ساده‌سازی موجود استفاده کرد. در نتیجه، مساله مورد نظر، یعنی ساده‌سازی چندضلعی قابل دید، را می‌توان با بکارگیری معیار جدید در الگوریتم‌های موجود حل کرد. درصورتی که در معیار ارائه شده از نسبت مساحت جابجاشده استفاده‌کنیم زمان اجرای الگوریتم برای حل مساله کمترین راس $O(n^۳)$ است که با همین مرتبه زمانی مساله ساده‌سازی چندضلعی قابل دید نقطه‌ای را نیز حل می‌کنیم. در صورتی که از نسبت فاصله به عنوان معیار خطا استفاده کنیم ابتدا یک الگوریتم با زمان اجرای $O(n^۲)$ برای حل آن ارائه می‌دهیم. سپس، یک الگوریتم کارا با زمان اجرای $O(n^{۴/۳})$ ارائه می‌کنیم که $V(q)$ را برای ناظر نقطه‌ای q ساده می‌کند. علاوه‌برآن، معیار ساده‌سازی ارائه شده را برای ساده‌سازی چندضلعی قابل دید ضعیف و قوی یک ناظر پاره‌خطی نیز تعمیم می‌دهیم و الگوریتم‌های کارایی برای حل این نوع ساده‌سازی ارائه می‌کنیم.

تنها کار قابل مقایسه توسط Buzer [۲۷] انجام شده است که به ساده‌سازی منحنی در هنگام نشان دادن بر روی یک صفحه نمایش پرداخته است. بااین وجود، در این روش نیز ساده‌سازی

مستقل از موقعیت ناظر و فقط بر اساس دقت صفحه نمایش صورت می‌گیرد.

در ادامه این فصل، در بخش ۸—۲ دو معیار ساده‌سازی وابسته به ناظر ارائه می‌شود. ویژگی‌های این معیارها، روش محاسبه خطا براساس آنها و الگوریتم‌های بدیهی ساده‌سازی براساس این معیارها (با مرتبه زمانی $O(n^3)$) نیز در این بخش بیان می‌شود. در بخش‌های ۸—۳ و ۸—۴ دو الگوریتم به ترتیب با مرتبه‌های زمانی $O(n^2)$ و $O(n^{4/3+\delta})$ برای ساده‌سازی براساس یکی از این معیارها بیان می‌شود. در بخش ۸—۵، معیار خطای ساده‌سازی چندضلعی قابل دید ضعیف و قوی ناظر پاره‌خطی و روش محاسبه آن بیان می‌شود. در بخش ۸—۶، الگوریتم کارای ساده‌سازی چندضلعی قابل دید ناظر پاره‌خطی بیان می‌شود.

۸-۲ ساده‌سازی وابسته به ناظر

در این فصل فقط گونه مقید مساله ساده‌سازی کمترین راس را بررسی می‌کنیم. فرض کنید $\mathcal{P}$ یک مسیر با دنباله رئوس $\langle p_0, p_1, p_2, \ldots, p_n \rangle$ است. هر زیر دنباله $\mathcal{Q} := \langle q_0, q_1, \ldots, q_{l+1} \rangle$ یک k-ساده‌سازی برای $\mathcal{P}$ است هرگاه رئوس q_0 و q_{l+1} به ترتیب برابر با رئوس p_0 و p_n باشند و $l \leq k$ باشد. در این ساده‌سازی، هر پاره‌خط q_iq_{i+1}، $(0 \leq i \leq l)$، ساده شده زیرمسیر $\mathcal{P}(s,t) := \langle p_s, p_{s+1}, \ldots, p_t \rangle$ از $\mathcal{P}$ است که $p_s = q_i$ و $p_t = q_{i+1}$. به عبارتی دیگر، مسیر $\mathcal{P}(s,t)$ به وسیله پاره‌خط q_iq_{i+1} ساده شده است. این نوع پاره‌خط‌ها را اتصال یا میان‌بر نیز می‌نامیم.

در این صورت، $\mathcal{Q}$ تقریبی از $\mathcal{P}$ است که با حافظه کمتری قابل نگهداری است. فرض کنید err تابع خطای معیار مورد استفاده برای تعیین میزان شباهت $\mathcal{P}$ و $\mathcal{Q}$ است. خطای این ساده‌سازی را بر اساس معیار err به دو صورت می‌توان تعریف کرد: در حالت اول، خطای ساده‌سازی $\mathcal{P}$ با $\mathcal{Q}$ را برابر با مجموع خطای میان‌برهای موجود در $\mathcal{Q}$ می‌گیریم که آن‌را با $SE_{err}(\mathcal{Q})$ نشان می‌دهیم. در حالت دیگر، خطای ساده‌سازی $\mathcal{P}$ با $\mathcal{Q}$ را برابر با بیشترین مقدار خطای میان‌برهای موجود در $\mathcal{Q}$ می‌گیریم و آن‌را با $ME_{err}(\mathcal{Q})$ نشان می‌دهیم. خطای میان‌بر q_iq_{i+1} از $\mathcal{Q}$ برای معیار خطای err با $err(q_iq_{i+1})$ نشان داده می‌شود و مقدار آن براساس تعریف معیار خطای err محاسبه می‌شود که معمولا وابسته به کاربرد تعیین می‌شود.

فاصله هاوس‌دورف که با err_h نشان داده می‌شود مهمترین معیار خطای استفاده شده در الگوریتم‌های ساده‌سازی موجود است. برای میان‌بر q_iq_{i+1} که ساده‌شده مسیر $\mathcal{P}(s,t)$ است، $err_h(q_iq_{i+1})$ برابر با بیشنیه فاصله اقلیدسی رئوس $\langle p_s, p_{s+1}, \ldots, p_t \rangle$ از پاره‌خط q_iq_{i+1} است. فاصله اقلیدسی نقطه p از یک پاره‌خط qr که با $d(p,qr)$ نشان داده می‌شود بصورت $|pt|$ تعریف می‌شود که t نقطه‌ای روی qr است که برای آن مقدار $|pt|$ کمینه است. به عنوان مثال، $err_h(p_rp_s)$ در شکل ۸—۱ برابر با $|p_kp_r|$ است که کمترین فاصله بین نقطه p_k و نقاط پاره‌خط p_rp_s است. همچنین، در این شکل، $err_h(p_sp_t)$ برابر با $d_o(p_m, p_sp_t)$ است که $d_o(p_m, p_sp_t)$ بصورت فاصله عمودی نقطه p_m از خط گذرنده از پاره‌خط p_sp_t تعریف می‌شود.

مساحت جابجا شده که با err_a نشان داده می‌شود یکی دیگر از معیارهای خطای مهم در الگوریتم‌های ساده‌سازی موجود است. براساس این معیار $err_a(q_iq_{i+1})$ برابر با مساحت بین مسیر $\mathcal{P}(s,t)$ و میان‌بر q_iq_{i+1} است که $\mathcal{P}(s,t)$ مسیر ساده‌شده بوسیله میان‌بر q_iq_{i+1} است.

در این معیارها فقط مسیر اولیه و مسیر ساده‌شده اهمیت دارند و به همین دلیل برای ساده‌سازی مرز ناحیه قابل دید از یک ناظر که موقعیت ناظر در آن اهمیت اساسی دارد مناسب نیستند. برای روشن شدن مطلب به مثال زیر توجه کنید.

فرض کنید $\mathcal{P} := \langle p_1, p_2, \ldots, p_n \rangle$ در شکل ۸—۱ چندضلعی قابل دید از ناظر نقطه‌ای q است. در این مثال، p_j نسبت به p_i بسیار نزدیک‌تر به q قرار دارد. حال فرض کنید که می‌خواهیم $\mathcal{P}$ را با حذف یک راس ساده کنیم و p_i و p_j تنها رئوس قابل حذف هستند. اگر راس p_j را حذف کنیم آنگاه موقعیت واقعی p_j نسبت به موقعیت آن در ساده‌سازی از دید q به اندازه $|p_jp'_j|$ تفاوت دارد و این جابجایی در فاصله $|p_jq|$ از ناظر اتفاق افتاده است. در مقابل، اگر راس p_i حذف شود آنگاه مقدار جابجایی نسبت به موقعیت اولیه برابر $|p_ip'_i|$ است و این جابجایی در فاصله $|p_iq|$ رخ می‌دهد. از آنجا که p_i نسبت به p_j در فاصله بسیار دورتری از q قرار دارد، ساده‌سازی به روش دوم، یعنی حذف p_i، از دید q مناسب‌تر است حتی اگر $|p_ip'_i|$ بصورت جزئی از $|p_jp'_j|$ بزرگ‌تر باشد. این در حالی است که بر اساس معیار خطای فاصله هاوس‌دورف باید p_i حذف شود. همین استدلال برای معیار مساحت جابجا شده نیز صدق می‌کند.

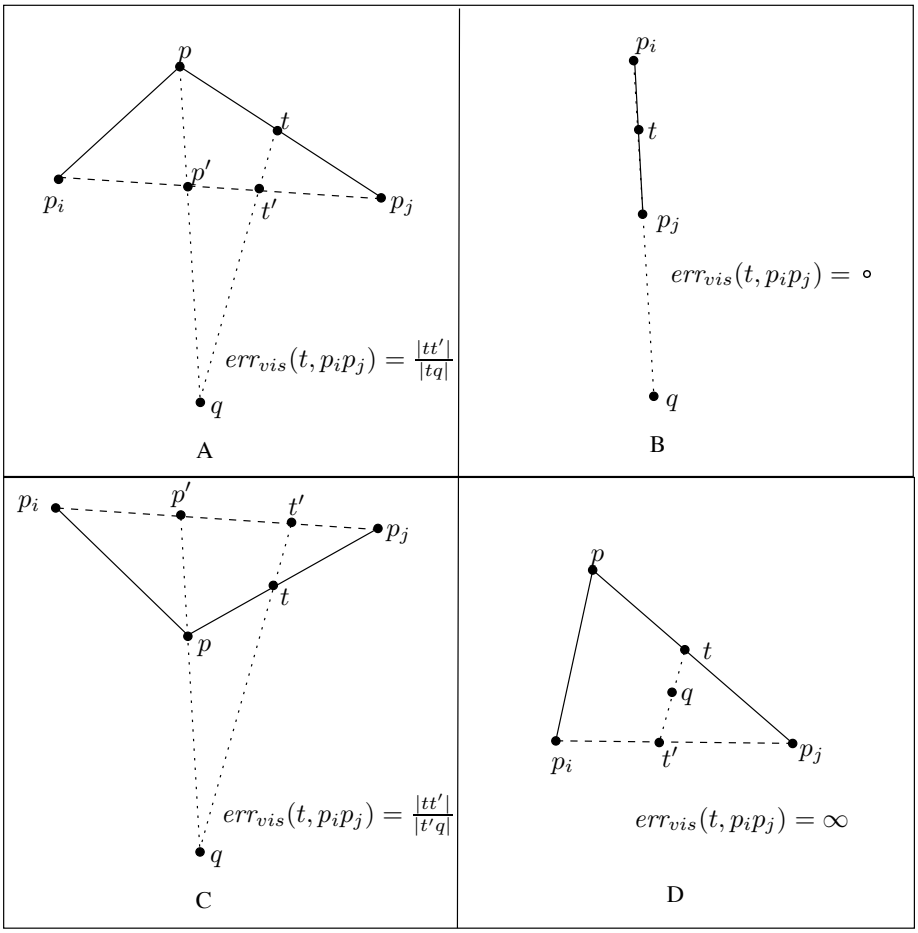
بنابراین، لازم است یک معیار خطای جدید تعریف شود که موقعیت ناظر را در محاسبات خود در نظر بگیرد. علاوه‌برآن، معیار جدید باید بسادگی قابل محاسبه و استفاده در الگوریتم‌های ساده‌سازی موجود باشد.

۸-۲-۱ تعریف معیار خطای وابسته به ناظر

فرض کنید می‌خواهیم مسیر p_ipp_j (نشان داده‌شده در قسمت A از شکل ۸—۲) را که بخشی از $V(q)$ است با پاره‌خط p_ip_j ساده کنیم. از دیدگاه q، نقطه p در این ساده‌سازی بر روی نقطه p' نگاشت می‌شود. همچنین سایر نقاط پاره‌خط‌های p_ip و pp_j بر روی نقاط متناظرشان در پاره‌خط‌های p_ip' و $p'p_j$ نگاشت می‌شوند. در این نوع ساده‌سازی، معیار خطای فاصله‌ای وابسته به ناظر را برای نقطه t از مسیر p_ipp_j و ناظر q بصورت $err_{vis(q)}(t, p_ip_j)$ نشان می‌دهیم. چنانچه ناظر q با توجه به متن مشخص باشد این خطا را بصورت ساده‌تر با $err_{vis}(t, p_ip_j)$ نشان می‌دهیم.

مقدار $err_{vis}(t, p_ip_j)$ را برابر با $\frac{|tt'|}{|tq|}$ تعریف می‌کنیم که t' محل تقاطع پاره‌خط‌های tq و p_ip_j است. این تعریف از خطا به این معنی است که در فاصله $|tq|$ از ناظر، فاصله مسیر اصلی با مسیر ساده شده برابر با $|tt'|$ است. این تعریف بصورت مشابه برای مسیرهای با بیش از یک راس میانی نیز تعریف می‌شود.

سه حالت وجود دارد که در تعریف بالا قرار نمی‌گیرند و باید تعریف را برای این حالات به صورت خاص بیان کرد. این سه حالت در قسمت‌های B، C و D از شکل ۸—۲ نشان داده شده‌اند.



شکل ۸-۲: معیار خطای ساده‌سازی وابسته به ناظر.

در حالت اول (قسمت B از شکل ۸—۲) رئوس p_i، p_j و q هم راستا هستند. می‌دانیم که این مسیر بخشی از $V(q)$ است و $V(q)$ یک چندضلعی ستاره‌ای است که q در هسته[۱] آن قرار دارد. بنابراین، در این حالت راس p و کلیه نقاط مسیر p_ipp_j نیز بر روی خط گذرنده از نقاط p_i، p_j و q قرار خواهند داشت. در نتیجه، پاره‌خط‌های tq و p_ip_j روی هم قرار می‌گیرند و نقطه یکتایی بر روی محل تقاطع آنها نخواهیم داشت. بدیهی است که در این حالت تقاوتی بین مسیر اصلی و مسیر ساده شده وجود ندارد و به همین دلیل خطای ساده‌سازی را برای این حالت برابر با صفر تعریف می‌کنیم.

حالت دیگر (قسمت C از شکل ۸—۲) مربوط به وضعیتی است که پاره‌خط‌های p_ip_j و tq همدیگر را قطع نمی‌کنند و در نتیجه نقطه t' وجود ندارد. در اینجا نیز با توجه به اینکه این مسیر بخشی از یک چندضلعی ستاره‌ای است و t بین دو راس p_i و p_j از این چندضلعی قرار دارد، امتداد خط گذرنده از tq حتما پاره‌خط p_ip_j را قطع می‌کند و این محل تقاطع بیرون $V(q)$ قرارمی‌گیرد. در این حالت نیز از نظر q نقطه t بر روی نقطه t' که محل این تقاطع است نگاشت می‌شود و مقدار $err_{vis}(t, p_ip_j)$ را برابر با $\frac{|tt'|}{|t'q|}$ تعریف می‌کنیم.

حالت سوم (قسمت D از شکل ۸—۲) مربوط به زمانی است که q روی پاره‌خط tt' قرار می‌گیرد. ما می‌خواهیم که مرز بیرونی ناحیه قابل دید را ساده کنیم و انتظار داریم که ناظر q همواره درون این ناحیه قرار داشته باشد. این درحالی است که با این ساده‌سازی q خارج از چندضلعی ساده شده قرار می‌گیرد. بنابراین، باید از این نوع ساده‌سازی‌ها جلوگیری شود. این حالت زمانی رخ می‌دهد که راسی مانند t روی مسیر $\mathcal{P}(i,j)$ وجود دارد که پاره‌خط‌های p_ip_j و tq همدیگر را قطع نمی‌کنند و p_ip_j درون $V(q)$ قرار دارد. در این حالت $err_{vis}(t, p_ip_j)$ را برابر با بینهایت تعیریف می‌کنیم و در نتیجه $err_{vis}(p_i, p_j)$ نیز، چنانچه در زیر می‌آید، بینهایت خواهد شد و این باعث می‌شود چنین پاره‌خطی در ساده‌سازی ظاهر نشود.

مطابق با تعریف بالا، مقدار err_{vis} برای هر نقطه از مرز $V(q)$ عددی بین صفر و یک یا برابر با بینهایت است.

با استفاده از تعریف بالا برای خطای یک نقطه، دو معیار فاصله هاوس‌دورف و مساحت جابجا شده را می‌توان برای حالت وابسته به ناظر تعریف کرد. خطای فاصله‌ای وابسته به ناظر نقطه‌ای برای پاره‌خط p_ip_j را که ساده شده مسیر $\mathcal{P}(i,j)$ است برابر با $\max_{t\in\mathcal{P}(i,j)}(err_{vis}(t, p_ip_j)$ تعریف می‌کنیم و با $err_{dvis(q)}(p_ip_j)$ یا به صورت ساده‌تر با $err_{dvis}(p_ip_j)$ نشان می‌دهیم. همچنین خطای مساحتی وابسته به ناظر را برای این پاره‌خط بصورت $\sum_{t\in\mathcal{P}(i,j)}(err_{vis}(t, p_ip_j)$ تعریف می‌کنیم و با $err_{avis(q)}(p_ip_j)$ یا $err_{avis}(p_ip_j)$ نشان می‌دهیم.

---

[۱] هسته یک چندضلعی مجموعه نقاطی از آن است که همه سطح چندضلعی را می‌بینند.

۸-۲-۲ معیار خطای فاصله‌ای وابسته به ناظر

رابطه نزدیکی بین معیار خطای فاصله‌ای وابسته به ناظر و مفهوم عرض نقاط وجود دارد. عرض یک مجموعه نقاط در امتداد داده شده $\overrightarrow{d}$ برابر با کمترین فاصله بین دو خط موازی در جهت $\overrightarrow{d}$ است که همه نقاط را در بر داشته باشد. فرض کنید $\mathcal{P}_L(i,j)$ ($\mathcal{P}_U(i,j)$) مجموعه رئوسی از مسیر $\mathcal{P}(i,j)$ است که در نیم‌صفحه بسته تعریف شده توسط خط گذرنده از نقاط p_i و p_j که شامل q است (نیست) قرار دارند. عرض مجموعه نقاط $\mathcal{P}_L(i,j)$ و $\mathcal{P}_U(i,j)$ را به ترتیب با $w_L(i,j)$ و $w_U(i,j)$ نشان می‌دهیم.

لم ۲۴ . برای مسیر $\mathcal{P}(i,j) = \langle p_i, p_{i+1}, \ldots, p_j \rangle$ از $V(q)$ داریم:

$$err_{dvis}(p_ip_j) = \max(\frac{w_U(i,j)}{d_o(q,p_ip_j)+w_U(i,j)}, \frac{w_L(i,j)}{d_o(q,p_ip_j)}).$$

اثبات. بر اساس قضیه تالس، $err_{vis}(p,p_ip_j)$ برای هر نقطه p از مسیر $\mathcal{P}(i,j)$ که در سمت دیگر p_ip_j نسبت به q قرار دارد برابر است با $\frac{d_o(p,p_ip_j)}{d_o(p,p_ip_j)+d_o(q,p_ip_j)}$. بنابراین، بیشترین خطای این نقاط برابر است با $\frac{w_U(i,j)}{d_o(q,p_ip_j)+w_U(i,j)}$. به طور مشابه، $err_{vis}(p,p_ip_j)$ برای هر نقطه p از مسیر $\mathcal{P}(i,j)$ که در همان سمتی از p_ip_j قرار دارند که q هست، برابر است با $\frac{d_o(p,p_ip_j)}{d_o(q,p_ip_j)}$ است و بیشینه این مقادیر برابر است با $\frac{w_L(i,j)}{d_o(q,p_ip_j)}$. □

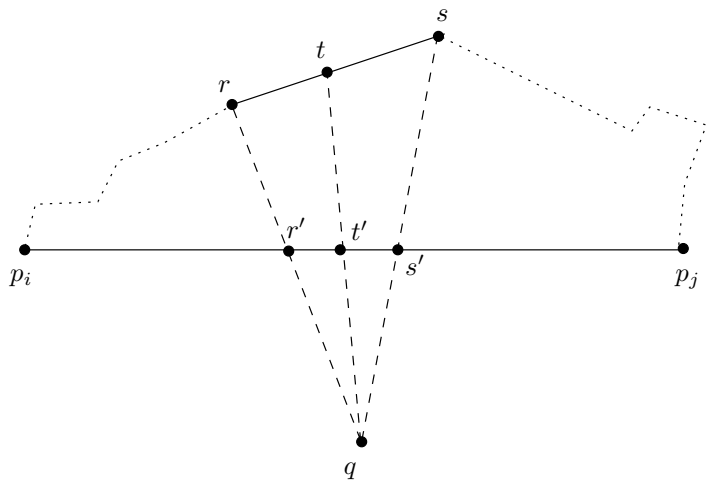
یک نتیجه مستقیم از این لم این است که $err_{dvis}(p_ip_j)$ مربوط به یک راس p_k از $\mathcal{P}(i,j)$ است. با استفاده از این نتیجه، بسادگی می‌توانیم خطای هر پاره‌خط p_ip_j را با محاسبه خطای راس‌های موجود در مسیر $\mathcal{P}(i,j)$ بدست بیاوریم.

۸-۲-۳ معیار خطای مساحتی وابسته به ناظر

خطای مساحتی وابسته به ناظر برای میان‌بر p_ip_j با مجموع خطای وابسته به ناظر نقاط مسیر $\mathcal{P}(i,j)$ برابر است. بنابراین، مقدار $err_{avis}(p_ip_j)$ را می‌توان با انتگرال‌گیری روی مقدار $err_{vis}(t,p_ip_j)$ به ازای همه مقادیر t از مسیر $\mathcal{P}(i,j)$ محاسبه کرد. فرض کنید پاره‌خط rs، که در شکل ۸—۳ نشان داده شده است، بخشی از مسیر $\mathcal{P}(i,j)$ است. خطای تجمعی مربوط به نقاط روی این پاره‌خط را با $err_{avis}(rs,p_ip_j)$ نشان می‌دهیم. بدون کاهش کلیت بحث، فرض کنید p_ip_j بر روی محور x قرار دارد. دراینصورت،

$$err_{avis}(rs,p_ip_j) = \int_{x_{r'}}^{x_{s'}} \frac{|tt'|}{|tq|}dx$$

که $x_{r'}$ و $x_{s'}$ به ترتیب مولفه x نقاط r' و s' هستند.



شکل ۸–۳: معیار خطای مساحتی وابسته به ناظر.

فرض کنید معادله خط گذرنده از rs به صورت $ax + by + c = ۰$ است و $x_q = m$ و $y_q = n$. همچنین، t روی هر دو خط گذرنده از پاره‌خط‌های rs و $t'q$ قرار دارد و طبق فرض بالا $x_{t'} = x$ و $y_{t'} = ۰$. درنتیجه، می‌توانیم x_t و y_t را به صورت زیر محاسبه کنیم:

$$\{y_t - n = \frac{n-۰}{m-x}(x_t - m), ax + by_t + c = ۰ \Rightarrow \{y_t = \frac{nc+anx}{ax-am-bn}, x_t = \frac{cm-cx-bnx}{ax-am-bn}.$$

بنابراین، مقدار $err_{avis}(rs, p_ip_j)$ برابر است با

$$err_{avis}(rs, p_ip_j) = \int_{x_{r'}}^{x_{s'}} \frac{|tt'|}{|tq|}dx = \int_{x_{r'}}^{x_{s'}} \frac{\sqrt{(x-x_t)^۲+(۰-y_t)^۲}}{\sqrt{(m-x_t)^۲+(n-y_t)^۲}}dx.$$

فرم بسته این انتگرال بصورت زیر بدست می‌آید:

$$\mathcal{F}(x) = \int \frac{|tt'|}{|tq|}dx = \frac{(t-x_t)\sqrt{(t-x_t)^۲+y_t^۲}+y_t^۲\ln(t-x_t+\sqrt{(t-x_t)^۲+y_t^۲})}{۲\sqrt{(m-x_t)^۲+(n-y_t)^۲}}.$$

در نتیجه مقدار $err_{avis}(rs, p_ip_j)$ از فرمول زیر محاسبه می‌شود:

$$err_{avis}(rs, p_ip_j) = \mathcal{F}(x_{s'}) - \mathcal{F}(x_{r'}).$$

بنابراین، $err_{avis}(p_ip_j)$ را می‌توان با محاسبه مقدار عبارت بالا به ازای هر پاره‌خط از مسیر $\mathcal{P}(i,j)$ بدست آورد. لازم به ذکر است که محاسباتی کاملا مشابه نیز برای حالتی که در قسمت C از شکل ۸–۲ نشان داده شده است انجام‌پذیر است. برای دو حالت دیگر نیز مقدار خطا ثابت و مشخص است.

۸-۲-۴ الگوریتم بدیهی ساده‌سازی

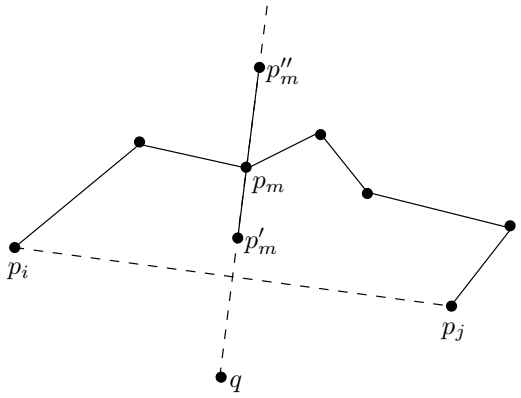
چنانچه گفته شد، الگوریتم‌های متعددی برای هر دو گونه مقید و بی‌قید مساله ساده‌سازی مسیر ارائه شده است. برخی از این الگوریتم‌ها مستقل از معیار خطا ارائه شده‌اند که قابل استفاده برای هر معیار خطای دلخواه برای ساده‌سازی هستند [۲۸، ۴۴، ۷۹، ۸۰، ۹۶، ۱۲۸]. به عنوان یک روش عمومی و بهینه، مساله ساده‌سازی با کمترین راس با ساختن یک گراف جهت‌دار و حل کوتاه‌ترین مسیر در این گراف حل می‌شود. در این روش، گراف جهت‌دار G_ϵ روی رئوس مسیر $\mathcal{P} = \langle p_0, p_1, \ldots, p_n \rangle$ ساخته می‌شود. در این گراف، به ازای هر راس p_i از $\mathcal{P}$ یک راس وجود دارد و یال جهت‌دار $\overrightarrow{p_i p_j}$ که $i < j$، به این گراف اضافه می‌شود اگر و تنها اگر $err(p_i p_j)$ بیشتر از مقدار خطای مجاز ϵ نباشد. سپس، کوتاه‌ترین مسیر $\mathcal{Q}$ از p_0 به p_n در این گراف جهت‌دار جواب بهینه مساله ساده‌سازی با کمترین راس مسیر $\mathcal{P}$ براساس هدف بهینه‌سازی $MErr(\mathcal{Q})$ است.

چنانچه بخواهیم $SErr(\mathcal{Q})$ بهینه باشد همه یال‌های $p_i p_j$ که $i < j$، به گراف جهت‌دار G_ϵ اضافه می‌شوند و وزن هر یال برابر با خطای مربوط به میان‌بر متناظر با آن در نظر گرفته می‌شود. دراین‌صورت، کوتاه‌ترین مسیر $\mathcal{Q}$ از p_0 به p_n جواب بهینه $SErr(\mathcal{Q})$ است.

برای هر دو حالت، زمان اجرای این الگوریتم، با فرض در دسترس بودن خطای همه پاره‌خط‌ها، برابر با $O(n^2)$ است. علاوه‌برآن، این روش مستقل از معیار خطای استفاده شده برای ساده‌سازی است. بنابراین، می‌توانیم هر دو معیار فاصله و مساحت وابسته به ناظر را برای ساده‌سازی چندضلعی قابل دید ناظر نقطه‌ای استفاده کنیم و پس از محاسبه خطای همه پاره‌خط‌ها، با این الگوریتم در زمان $O(n^2)$ عمل ساده‌سازی را انجام دهیم. تعداد پاره‌خط‌های $p_i p_j$ برابر با $O(n^2)$ است. به صورت بدیهی می‌توانیم خطای $err_{avis}(p_i p_j)$ و $err_{dvis}(p_i p_j)$ را برای هر پاره‌خط $p_i p_j$ در زمان $O(n)$ محاسبه کنیم. بنابراین، کل زمان اجرای ساده‌سازی برابر با $O(n^3)$ خواهد شد. در بخش‌های بعدی دو روش برای حل این مساله برای معیار خطای فاصله وابسته به ناظر و با زمان اجرای کمتر از $O(n^3)$ ارائه می‌کنیم.

۸-۳ بهبود الگوریتم عمومی ساده‌سازی برای معیار فاصله‌ای وابسته به ناظر

در الگوریتم عمومی که در بخش ۸—۲—۴ بیان شد، ساده‌سازی برای معیار فاصله‌ای وابسته به ناظر در زمان $O(n^3)$ انجام می‌شود. این مرتبه زمانی مربوط به زمان لازم برای محاسبه $err_{dvis}(p_i p_j)$ است. پس از این محاسبه، الگوریتم اصلی (محاسبه کوتاه‌ترین مسیر) در زمان $O(n^2)$ اجرا می‌شود. در این بخش، روشی برای محاسبه $err_{dvis}(p_i p_j)$ برای همه پاره‌خط‌های $p_i p_j$ که ممکن است در ساده‌سازی ظاهر شوند ارائه می‌کنیم که در زمان $O(n^2)$ همه این خطاها را محاسبه می‌کند. در



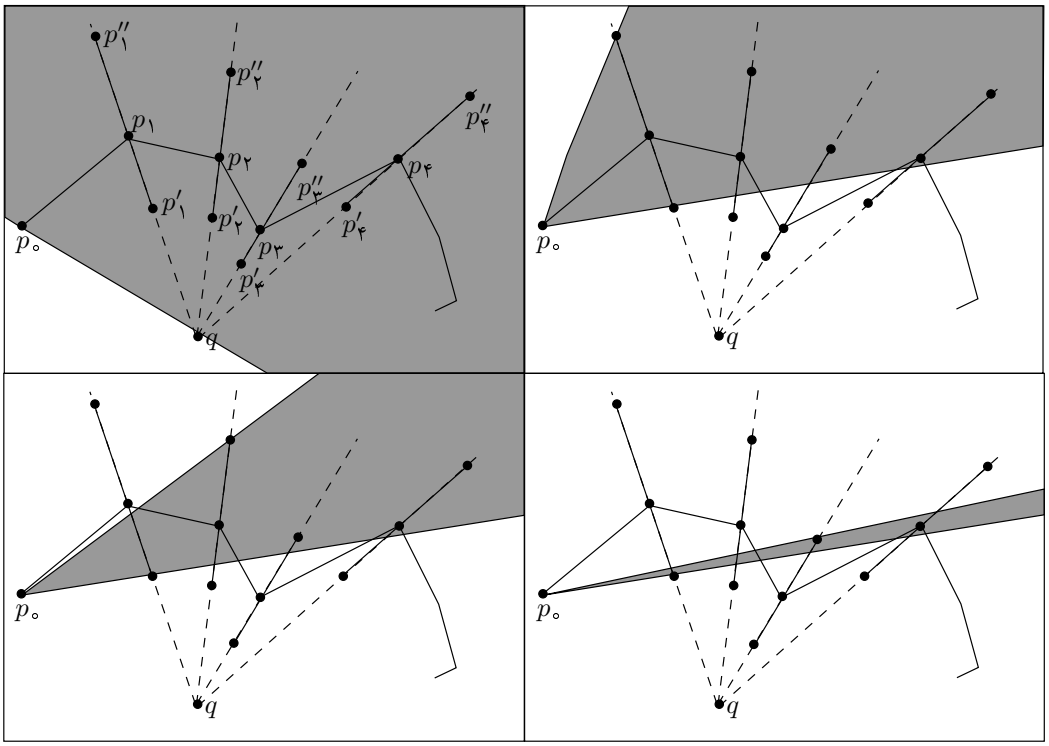
شکل ۸–۴: ناحیه نوسان برای ناظر نقطه‌ای.

نتیجه، الگوریتم ساده‌سازی ارائه شده در بخش ۸—۲—۴، در زمان $O(n^2)$، ساده‌سازی بر اساس معیار فاصله‌ای وابسته به ناظر را انجام می‌دهد.

پاره‌خط p_ip_j فقط در صورتی در ساده‌سازی با کمترین راس مسیر $\mathcal{P}$ می‌تواند ظاهر شود که برای همه نقاط p_m که $i < m < j$، شرط $err_{vis}(p_m, p_ip_j) \leq \epsilon$ برقرار باشد که ϵ مقدار خطای مجاز داده شده برای مساله کمترین راس است. برای برآورده کردن این شرط، p_ip_j باید پاره‌خط $p'_mp''_m$ را قطع کند که طبق شکل ۸—۴، p'_m نقطه‌ای روی پاره‌خط p_mq است بطوریکه $|p'_mp_m| = \epsilon|p_mq|$ و p''_m نقطه‌ای روی خط گذرنده از نقاط q و p_m و در سمت مقابل p_m نسبت به p'_m است بطوریکه $|p''_mp_m| = \epsilon|p''_mq|$. علاوه‌برآن، افزودن پاره‌خط p_ip_j به ساده‌سازی نباید باعث شود که q خارج ساده‌سازی قرار گیرد. برای ارضای این شرط، لازم است که p_j درون نیم‌صفحه‌ای قرار گیرد که توسط خط گذرنده از p_iq تعریف می‌شود و شامل p_{i+1} است. این نیم‌صفحه را برای نقطه p_i از مسیر $\mathcal{P}$ و ناظر q با $hp(p_i, \mathcal{P}, q)$ نشان می‌دهیم.

به عبارت دیگر، یال $\overrightarrow{p_ip_j}$ به گراف جهت‌دار G_ϵ مربوط به الگوریتم عمومی ساده‌سازی اضافه می‌شود اگر و تنها اگر p_ip_j همه پاره‌خط‌های $p'_mp''_m$ را برای نقاط p_m، $i < m < j$، قطع کند و q درون ساده‌سازی شامل p_ip_j قرار گیرد. پاره‌خط $p'_mp''_m$ را ناحیه نوسان نقطه p_m نسبت به ناظر q و خطای آستانه ϵ می‌نامیم.

بنابراین، برای تصمیم‌گیری در مورد پاره‌خط p_ip_j که آیا باید به گراف جهت‌دار G_ϵ افزوده شود یا خیر، باید ناحیه نوسان همه رئوس $\mathcal{P}(i, j)$ که بین p_i و p_j قرار دارند، بررسی شوند. سپس، برای تصمیم‌گیری در مورد p_ip_{j+1} می‌توانیم از اطلاعات بدست آمده در زمان تصمیم‌گیری در مورد p_ip_j استفاده کنیم. برای این منظور، ناحیه مجاز راس p_i را که با PR_{p_i} نشان داده می‌شود، بصورت زیر تعریف می‌کنیم. ناحیه مجاز PR_{p_i} بخشی از صفحه است که هنگام تصمیم‌گیری در مورد p_ip_j، این یال به G_ϵ اضافه می‌شود اگر و تنها اگر p_j درون ناحیه PR_{p_i} باشد. بدیهی است که مقدار اولیه PR_{p_i} برابر است با $hp(p_i, \mathcal{P}, q)$. دلیل آن این است که $err_{dvis}(p_ip_{i+1}) = 0$ و صرف‌نظر از



شکل ۸–۵: ناحیه خاکستری، تغییرات ناحیه مجاز راس p_0 را هنگام پردازش راس‌های p_1 تا p_4 نشان می‌دهد.

موقعیت p_{i+1}، یال $\overrightarrow{p_ip_{i+1}}$ همواره به G_ϵ اضافه می‌شود. سپس، $\overrightarrow{p_ip_{i+2}}$ به G_ϵ اضافه می‌شود اگر و تنها اگر p_{i+2} درون گوشه $\angle p'_{i+1}p_ip''_{i+1}$ باشد که $\angle abc$ گوشه تعریف شده بوسیله یال‌های $\overrightarrow{ba}$ و $\overrightarrow{bc}$ است که شامل پاره خط ac است و $p'_{i+1}p''_{i+1}$ ناحیه نوسان نقطه p_{i+1} است. بنابراین، پس از بررسی راس p_{i+1}، باید RP_{p_i} را بهنگام‌سازی کرد و مقدار آن را برابر با $\angle p'_{i+1}p_ip''_{i+1}$ گرفت. دراین‌صورت، هنگامی که راس p_{i+2} بررسی می‌شود، PR_{p_i} ویژگی مورد نظر را دارد و یال $\overrightarrow{p_ip_{i+2}}$ به G_ϵ اضافه می‌شود اگر و تنها اگر p_{i+2} درون ناحیه PR_{p_i} قرار داشته باشد. در شکل ۸—۵، تغییرات ناحیه مجاز راس p_0 هنگام تصمیم‌گیری در مورد پاره‌خط‌های p_0p_k، $(0 < k < 5)$، نشان داده شده است. طبق این شکل، یال‌های $\overrightarrow{p_0p_1}$، $\overrightarrow{p_0p_2}$ و $\overrightarrow{p_0p_4}$ به G_ϵ اضافه می‌شوند ولی $\overrightarrow{p_0p_3}$ به G_ϵ اضافه نمی‌شود. با استفاده از این روش می‌توانیم گراف G_ϵ را در زمان $O(n^2)$ بسازیم. شبه کد مربوط به محاسبه G_ϵ در شکل ۸—۶ نشان داده شده است.

بنابراین، ساده‌سازی براساس معیار خطای فاصله‌ای وابسته به ناظر را می‌توان با استفاده از الگوریتم عمومی و بهینه موجود و بدون افزایش زمان یا حافظه مورد نیاز آن، حل کرد.

```
BuildDAG(P, ε)
    ▷ P is the input path of n vertices and ε is the error threshold
 1  for i = 1 to n − 1
 2    do
 3      PR_i := hp(p_i, P, q)
 4      for j = i + 1 to n
 5        do
 6          if p_j ∈ PR_i
 7            then add p_i p_j (directed) to G_ε
 8          PR_i = PR_i ∩ ∠p'_j p_i p''_j
 9  return G_ε
```

شکل ۸-۶: الگوریتم ساختن G_ϵ مربوط به الگوریتم عمومی ساده‌سازی برای ساده‌سازی چندضلعی‌قابل دید ناظر نقطه‌ای براساس معیار فاصله‌ای وابسته به ناظر.

قضیه ۱۸ . چندضلعی قابل دید ناظر نقطه‌ای واقع در یک دامنه چندضلعی گونه را می‌توان براساس معیار فاصله وابسته به ناظر با کمترین راس در زمان $O(n^2)$ ساده کرد که n تعداد رئوس موجود در چندضلعی قابل دید است.

۸-۴ الگوریتم کارای ساده‌سازی چندضلعی قابل دید نقطه‌ای

در صورتی که به دنبال الگوریتم‌هایی با مرتبه زمانی کمتر از $O(n^2)$ باشیم، گراف G_ϵ را نباید به طور کامل و صریح محاسبه کنیم. راه‌حل غلبه بر این مشکل تهیه نمایش فشرده‌ای از G_ϵ است که اندازه آن کمتر از $O(n^2)$ باشد. Agarwal و Varadarjan [۷] نخستین الگوریتم ساده‌سازی با مرتبه زمانی $O(n^{4/3+\delta})$ را برای حل مساله ساده‌سازی با کمترین راس با استفاده از معیار خطای فاصله هاوس‌دورف در L_1 ارائه دادند. این الگوریتم مبتنی بر فشرده‌سازی گراف G_ϵ با استفاده از پوشش خوشه‌ای[۲] است. ایده فشرده‌سازی گراف با استفاده از پوشش خوشه‌ای توسط Feder و Motaven [۴۶] مطرح شد که با استفاده از آن مرتبه زمانی اجرای تعدادی از الگوریتم‌های گراف را بهبود دادند.

در این بخش، ابتدا روش تهیه پوشش خوشه‌ای را که در [۷] برای نمایش فشرده گراف G_ϵ ارائه شده است، بیان می‌کنیم. همچنین الگوریتم عمومی ساده‌سازی بر اساس نمایش فشرده G_ϵ را بیان می‌کنیم. سپس روش محاسبه نمایش فشرده G_ϵ را برای چندضلعی قابل دید ناظر نقطه‌ای بیان می‌کنیم و نشان می‌دهیم که چگونه این کار باعث بهبود سرعت ساده‌سازی می‌شود.

[۲] Clique Cover

گراف جهت دار $G = (V, E)$ و مجموعه $\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_l = (V_l, E_l)\}$ را درنظر بگیرید که هر G_i یک زیرگراف از G است. $\mathcal{G}$ یک پوشش خوشه‌ای برای G است اگر سه شرط زیر برقرار باشد:

۱) هر گراف G_i یک گراف کامل دوبخشی است و اگر V_{i_1} و V_{i_2} دو دسته رئوس این گراف باشند آنگاه جهت هر یال G_i باید از یک راس از V_{i_1} به راسی از V_{i_2} باشد.

۲) $E = E_1 \cup E_2 \ldots \cup E_l$.

۳) به ازای هر i و j که $1 \leq i \leq l$، $1 \leq j \leq l$ و $i \neq j$، $E_i \cap E_j = \Phi$.

با توجه به اینکه هر گراف G_i یک خوشه دو بخشی است، می‌توانیم آن را بصورت فشرده‌تر و فقط با تعیین دو دسته راس‌های V_{i_1} و V_{i_2} نگهداری کنیم. در نتیجه، حافظه مورد نیاز آن $O(|V_i|)$ است و نیازی به نگهداری یالهای G_i نیست. اندازه پوشش خوشه‌ای $\mathcal{G}$ را با $|\mathcal{G}|$ نشان می‌دهیم و مقدار آن را برابر با $\sum_{i=1}^{l} |V_i|$ تعریف می‌کنیم. در واقع حافظه مورد نیاز برای نگهداری فشرده G برابر با $|\mathcal{G}|$ است.

با داشتن شکل فشرده $\mathcal{G}$ از گراف G، کوتاه‌ترین مسیر بین دو راس از G را می توان در زمان $O(|\mathcal{G}| + |V|)$ محاسبه کرد.

لم ۲۵ . اگر $\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_l = (V_l, E_l)\}$ یک پوشش خوشه‌ای برای گراف جهت‌دار G باشد آنگاه کوتاه‌ترین مسیر بین دو راس u و v از G را می توان در زمان $O(|\mathcal{G}| + |V|)$ محاسبه کرد.

اثبات. گراف جدید H با مجموعه رئوس $V \cup \{g_1, g_2, \ldots, g_l\}$ را تعریف می‌کنیم که هر g_i متناظر با G_i است. به ازای هر یال جهت‌دار $\overrightarrow{uv}$ از G، اگر $\overrightarrow{uv}$ متعلق به G_i باشد، یال‌های $\overrightarrow{ug_i}$ و $\overrightarrow{g_iv}$ به گراف جدید H اضافه می‌شوند. در نتیجه گراف H دارای $O(l + |V|)$ راس و $O(|\mathcal{G}|)$ یال است. برای محاسبه کوتاه‌ترین مسیر بین u و v از G، کوتاه‌ترین مسیر بین u و v در H محاسبه می‌شود. اگر این مسیر بصورت $\langle u_1 = u, g_1, u_2, g_2, \ldots, g_k, u_{k+1} = v\rangle$ باشد، آنگاه مسیر $\langle u_1, u_2, \ldots, u_{k+1}\rangle$ کوتاه‌ترین مسیر بین u و v در G است. اثبات صحت این ادعا سرراست است و بدیهی است که این محاسبه در زمان $O(|\mathcal{G}| + |V|)$ قابل انجام است. □

حال الگوریتمی را که در [۷] برای محاسبه $\mathcal{G}$ براساس معیار خطای L_1 و یکنواخت[۳] ارائه شده است، برای چندضلعی قابل دید ناظر نقطه‌ای و معیار خطای فاصله‌ای وابسته به ناظر تعمیم می‌دهیم. دراین روش $\mathcal{G}$ به‌صورت تقسیم و حل[۴] ساخته می‌شود. فرض کنید $V(q) = C = \langle p_1, p_2, \ldots, p_n\rangle$ چندضلعی قابل دید مورد نظر است و $G_\epsilon(C)$ گراف جهت‌دار متناظر با آن (طبق الگوریتم عمومی ساده‌سازی) و $\mathcal{G}$ پوشش خوشه‌ای $G_\epsilon(C)$ است. فرض کنید

[۳] Uniform

[۴] Divide-and-Conquer

$C_۱ = \langle p_۱, p_۲, \ldots, p_{\lfloor \frac{n}{۲} \rfloor} \rangle$ و $C_۲ = \langle p_{\lfloor \frac{n}{۲} \rfloor + ۱}, \ldots, p_n \rangle$ دو بخش C باشند. $\mathcal{G}_۱$ و $\mathcal{G}_۲$ به ترتیب پوشش خوشه‌ای $G_\epsilon(C_۱)$ و $G_\epsilon(C_۲)$ هستند که بصورت بازگشتی محاسبه می‌شوند. در مرحله ادغام، پوشش خوشه‌ای $\mathcal{G}_{۱۲}$ را برای یال‌های

$$E_{۱۲} = \{\overrightarrow{p_i p_j} | \overrightarrow{p_i p_j} \in G_\epsilon(C), p_i \in C_۱, p_j \in C_۲\}$$

بدست می‌آوریم. بنابراین، $\mathcal{G}_۱$ و $\mathcal{G}_۲$ به ترتیب شامل همه یال‌هایی هستند که دو راس آنها در $C_۱$ و $C_۲$ قرار دارد و $\mathcal{G}_{۱۲}$ شامل همه یال‌هایی است که یک سر آنها در $C_۱$ و سر دیگر آنها در $C_۲$ است. درنتیجه، $\mathcal{G} = \mathcal{G}_۱ \cup \mathcal{G}_۲ \cup \mathcal{G}_{۱۲}$ یک پوشش خوشه‌ای برای $G_\epsilon(C)$ است. با توجه به اینکه $\mathcal{G}_۱$ و $\mathcal{G}_۲$ به صورت بازگشتی محاسبه می‌شوند، کافی است نحوه محاسبه $\mathcal{G}_{۱۲}$ را بیان کنیم.

در بخش ۸—۳، PR_{p_i} را به این ترتیب تعریف کردیم که مقدار آن ابتدا برابر است با یک نیم‌صفحه و با پردازش رئوس $p_{i+۱}$، $p_{i+۲}$ و ... مقدار آن به تدریج بهنگام‌سازی می‌شود. مسیرهای $C_۱ = \langle p_۱, p_۲, \ldots, p_{\lfloor \frac{n}{۲} \rfloor} \rangle$ و $\overline{C_۲} = \langle p_n, p_{n-۱}, \ldots, p_{\lfloor \frac{n}{۲} \rfloor + ۱} \rangle$ و رئوس $p_i \in C_۱$ و $p_j \in \overline{C_۲}$ را در نظر بگیرید. $cone(p_i)$ را برابر با مقدار PR_{p_i} پس از پردازش راس $p_{\lfloor \frac{n}{۲} \rfloor}$ و $cone(p_j)$ را برابر با مقدار PR_{p_j} پس از پردازش راس $p_{\lfloor \frac{n}{۲} \rfloor + ۱}$ در نظر می‌گیریم. بدین ترتیب، لم زیر را داریم.

لم ۲۶ . $\overrightarrow{p_i p_j} \in E_{۱۲}$ اگر و تنها اگر $p_j \in cone(p_i)$ و $p_i \in cone(p_j)$.

اثبات. در بخش ۸—۳ نشان دادیم که $\overrightarrow{p_i p_j} \in G_\epsilon$ اگر و تنها اگر $p_i p_j$ همه پاره‌خط‌های $p'_m p''_m$، $i < m < j$، را قطع کند. بدیهی است اگر $p_i p_j$ همه این پاره‌خط‌ها را قطع کند آنگاه $p_j \in cone(p_i)$ و $p_i \in cone(p_j)$. از طرف دیگر، اگر $p_j \in cone(p_i)$ و $p_i \in cone(p_j)$، $p_i p_j$ همه پاره‌خط‌های $p'_m p''_m$، $i < m < j$، را قطع خواهد کرد و در نتیجه متعلق به G_ϵ و نیز متعلق به $E_{۱۲}$ است. □

فرض کنید نیم‌خط‌های $p_i l$ و $p_i l'$ نیم‌خط‌های تعریف کننده $cone(p_i)$ هستند و α زاویه مربوط به $cone(p_i)$ و در نتیجه زاویه محدب بین این نیم‌خط‌ها است. $dwed(p_i)$ را برابر با ناحیه محدود شده بین دو خط گذرنده از $p_i l$ و $p_i l'$ و با زاویه α تعریف می‌کنیم. درنتیجه، $dwed(p_i)$ شامل $cone(p_i)$ و قرینه آن نسبت به p_i است. طبق این تعریف لم زیر برقرار است.

لم ۲۷ . عبارات زیر هم‌ارز هستند.

۱) $\overrightarrow{p_i p_j} \in E_{۱۲}$.

۲) $p_j \in cone(p_i)$ و $p_i \in cone(p_j)$.

۳) $p_j \in dwed(p_i)$، $p_i \in dwed(p_j)$، $p_j \in hp(p_i, C_۱, q)$ و $p_i \in hp(p_j, \overline{C_۲}, q)$.

اثبات. اثبات ۲ $\Leftrightarrow$ ۱ در لم ۲۶ بیان شد. اثبات ۳ $\Rightarrow$ ۲ نیز با توجه به تعریف $cone$ و $dwed$ بدیهی است. بنابراین کافی‌است ۲ $\Rightarrow$ ۳ را اثبات کنیم.

طبق تعریف $cone$ و $dwed$ داریم:

$$cone(p_i) = hp(p_i, C_1, q) \cap dwed(p_i),$$
$$cone(p_j) = hp(p_j, \overline{C_2}, q) \cap dwed(p_j).$$

از این رابطه‌ها نتیجه **۲** $\Rightarrow$ **۳** بدست می‌آید. □

با توجه به لم بالا، مجموعه یال‌های E_{12} را بصورت زیر می‌توان نوشت:

$$\begin{aligned} E'_{12} &= \{\overrightarrow{p_ip_j} | p_i \in C_1, p_j \in \overline{C_2}, p_j \in hp(p_i, C_1, q)\}, \\ E''_{12} &= \{\overrightarrow{p_ip_j} | \overrightarrow{p_ip_j} \in E'_{12}, p_i \in hp(p_j, \overline{C_2}, q)\}, \\ E_{12} &= \{\overrightarrow{p_ip_j} | \overrightarrow{p_ip_j} \in E''_{12}, p_i \in dwed(p_j), p_j \in dwed(p_i)\}. \end{aligned}$$

با داشتن $cone(p_i)$ مربوط به نقاط $p_i \in C_1$ و $p_i \in \overline{C_2}$، که روش محاسبه آنها را در لم **۲۸** خواهد آمد، پوشش خوشه‌ای $\mathcal{G}_{12}$ را بصورت زیر محاسبه می‌کنیم.

ابتدا پوشش خوشه‌ای $\mathcal{G}'_{12}$ را برای یال‌های E'_{12} محاسبه می‌کنیم. سپس، به ازای هر خوشه $(A, B) \in \mathcal{G}'_{12}$ یک پوشش خوشه‌ای از یال‌های $\{(a, b) | (a, b) \in A \times B, (a, b) \in E''_{12}\}$ بدست می‌آوریم که منجر به یک پوشش خوشه‌ای $\mathcal{G}''_{12}$ برای یال‌های E''_{12} می‌شود. در نهایت، برای هر خوشه $(A', B') \in \mathcal{G}''_{12}$ یک پوشش خوشه‌ای برای مجموعه یال‌های $\{(a, b) | (a, b) \in A' \times B', a \in dwed(b), b \in dwed(a)\}$ بدست می‌آوریم. بدیهی است که پوشش خوشه‌ای اخیر یک پوشش خوشه‌ای برای E_{12} است.

لم ۲۸ . $cone(p_i)$ را می‌توان برای همه رئوس $p_i \in C_1$ و نیز $p_i \in \overline{C_2}$ در زمان $O(n \log n)$ محاسبه کرد.

اثبات. فرض کنید $p_i \in C_1$ و بصورت استقرایی، مقدار $cone(p_k)$ را برای $i \leq k \leq \lfloor \frac{n}{4} \rfloor$ محاسبه کرده‌ایم. L_i را برابر با مرز زیرین پوش محدب نقاط $\{p''_i, p''_{i+1}, \ldots, p''_{\lfloor \frac{n}{4} \rfloor}\}$ (نسبت به خط گذرنده از $p_ip_{\lfloor \frac{n}{4} \rfloor}$) و U_i را برابر با مرز بالایی پوش محدب نقاط $\{p'_i, p'_{i+1}, \ldots, p'_{\lfloor \frac{n}{4} \rfloor}\}$ تعریف می‌کنیم. فرض می‌کنیم که مقدار L_i و U_i را نیز بصورت استقرایی محاسبه کرده‌ایم.

برای راس p_{i-1}، مماس‌هایی را که از نقطه p_{i-1} به L_i و U_i رسم می‌شوند بدست می‌آوریم. چنانچه مماس بر L_i در زیر مماس بر U_i (نسبت به خط گذرنده از $p_{i-1}p_{\lfloor \frac{n}{4} \rfloor}$) قرار بگیرد، $cone(p_{i-1})$ تهی است. درغیراینصورت، $cone(p_{i-1})$ برابر با ناحیه بین این دو مماس است. L_{i-1} و U_{i-1} را با استفاده از یکی از روش‌های موجود [۹۵، ۱۱۳] در زمان $O(\log n)$ محاسبه می‌کنیم. مماسهای U_i و L_i را نیز می‌توانیم در زمان $O(\log n)$ تعیین کنیم. درنتیجه، زمان لازم برای محاسبه $cone(p_i)$ برای همه نقاط $p_i \in C_1$ برابر است با $O(n \log n)$. برای نقاط $p_i \in \overline{C_2}$ نیز همین استدلال صدق می‌کند. □

لم ۲۹ . برای $E'_{۱۲}$ می‌توان یک پوشش خوشه‌ای با اندازه $O(n\log n)$ و در زمان $O(n\log n)$ ساخت.

اثبات. عناصر $p_j \in \overline{C_۱}$ را در یک داده‌ساختار بگونه‌ای قرار می‌دهیم که بتوانیم پرس‌وجوی زیر را بصورت کارایی انجام دهیم: برای هر نقطه $p_i \in C_۱$، همه نقاط $p_j \in \overline{C_۱}$ را که در $hp(p_i, C_۱, q)$ قرار می‌گیرند محاسبه کنید.

برای این‌کار، یک داده‌ساختار جستجوی دودویی روی نقاط $\overline{C_۲}$ ایجاد می‌کنیم که جواب مربوط به پرس‌وجوی p_i را بصورت اجتماع $O(\log n)$ زیرمجموعه کانونی و جدا از هم بدست می‌آورد. اندازه همه مجموعه‌های کانونی در این داده‌ساختار برابر است با $O(n)$. با داشتن این داده‌ساختار، به ازای هر $p_i \in C_۱$، پرس‌وجوی مربوط را اجرا می‌کنیم و به ازای هر مجموعه کانونی B_i از این داده‌ساختار، مجموعه A_i شامل نقاطی از $C_۱$ که B_i جزء جواب پرس‌وجوی آنها بوده است را بدست می‌آوریم. اگر A_i تهی نباشد، آنگاه (A_i, B_i) را به عنوان یکی از اعضای پوشش خوشه‌ای $\mathcal{G}'_{۱۲}$ انتخاب می‌کنیم. اندازه مجموعه‌های کانونی B_i برابر است با $O(n)$ و هر یک از عناصر $C_۱$ نیز حداکثر در $O(\log n)$ مجموعه از A_iها اضافه می‌شوند. در نتیجه، اندازه پوشش خوشه‌ای بدست آمده $O(n\log n)$ است و در زمان $O(n\log n)$ نیز قابل محاسبه است. □

لم ۳۰ . برای $E''_{۱۲}$ می‌توان یک پوشش خوشه‌ای با اندازه $O(n\log^۲ n)$ و در زمان $O(n\log^۲ n)$ ساخت.

اثبات. طبق رابطه بین $E'_{۱۲}$، $E''_{۱۲}$، به ازای هر یک از اعضای (A_i, B_i) از پوشش خوشه‌ای بدست آمده برای $E'_{۱۲}$ مطابق با لم بالا، یک پوشش خوشه‌ای روی زیرمجموعه‌ای از یال‌های $A_i \times B_i$ ایجاد می‌کنیم که فقط شامل یال‌های موجود در $E''_{۱۲}$ باشد.

برای این‌کار، خوشه $(A_i, B_i) \in \mathcal{G}'_{۱۲}$ را در نظر بگیرید. داده‌ساختار جستجوی بازه بیان شده در لم ۲۹ را برای اعضای مجموعه A_i بگونه‌ای می‌سازیم که بتوانیم پرس‌وجوی زیر را جواب دهیم. به ازای هر $p_j \in B_i$، مجموعه نقاط $p_i \in A_i$ را که $p_i \in hp(p_j, \overline{C_۲}, q)$ بدست آورید. همانند لم ۲۹، یک پوشش خوشه‌ای برای این زیرمجموعه از یال‌های $A_i \times B_i$ بدست می‌آوریم که اندازه آن برابر $O(|A_i \cup B_i| \log(|A_i \cup B_i|))$ است و در همین مرتبه زمانی نیز قابل ساخت است.

اگر این‌کار را به ازای همه خوشه‌های (A_i, B_i) انجام دهیم، آنگاه اندازه کل پوشش‌های خوشه‌ای بدست آمده برابر با $O(n\log^۲ n)$ خواهد شد و زمان محاسبه آنها نیز $O(n\log^۲ n)$ است. بدیهی است که هر یال موجود در این پوشش خوشه‌ای متعلق به $E''_{۱۲}$ است و هر یال متعلق به $E''_{۱۲}$ نیز در این پوشش خوشه‌ای وجود دارد. □

ازطرفی، در[۷] یک روش برای تهیه پوشش خوشه‌ای مربوط به مجموعه یال های $E = \{(a,b) | a \in A, b \in B, a \in dwed(b), b \in dwed(a)\}$ ارائه شده است. این روش مبتنی بر نگاشت خطوط $dwed(p)$ به فضای دوگان و محاسبه تقاطع پاره‌خط‌ها در این فضای دوگان است. زمان لازم

برای تهیه این پوشش خوشه‌ای و اندازه پوشش خوشه‌ای حاصل برابر با $O(n^{۴/۳+\delta})$ است که n تعداد نقاط مجموعه‌های A و B است.

با استفاده از این روش، و داشتن پوشش خوشه‌ای مربوط به یال‌های $E''_{۱۲}$، پوشش خوشه‌ای $E_{۱۲}$ را بصورت زیر می‌سازیم. به ازای هر یک از اعضای (A'_i, B'_i) از پوشش خوشه‌ای بدست آمده برای $E''_{۱۲}$ مطابق با لم ۳۰، یک پوشش خوشه‌ای روی زیرمجموعه‌ای از یال‌های $A'_i \times B'_i$ مطابق با روش ارائه شده در[۷] ایجاد می‌کنیم که فقط شامل یال‌های موجود در $E_{۱۲}$ باشد. بدیهی است که زمان تهیه و اندازه این پوشش خوشه‌ای برای کلیه اعضای پوشش خوشه‌ای مربوط به $E''_{۱۲}$ برابر است با $O(n^{۴/۳+\delta})$.

بنابراین، زمان اجرا و حافظه مصرفی الگوریتم تقسیم و حل محاسبه پوشش خوشه‌ای G_ϵ بصورت زیر خواهد بود:

$$\begin{aligned} S(n) &\leq ۲S(n/۲) + c.n^{۴/۳+\delta} \\ &= O(n^{۴/۳+\delta}) \end{aligned}$$

لم ۳۱ . یک پوشش خوشه‌ای برای G_ϵ مربوط به چندضلعی قابل دید نقطه‌ای و معیار خطای فاصله‌ای وابسته به ناظر با اندازه $O(n^{۴/۳+\delta})$ وجود دارد که در زمان $O(n^{۴/۳+\delta})$ قابل ساخت است.

با ترکیب لم بالا و نتیجه مربوط به لم ۲۵، می‌توان ساده‌سازی $V(q)$ را در زمان کمتر از درجه دو انجام داد.

قضیه ۱۹ . چندضلعی قابل دید یک ناظر نقطه‌ای واقع در یک دامنه چندضلعی گونه را می‌توان با صرف زمان و حافظه $O(n^{۴/۳+\delta})$ بر اساس معیار فاصله‌ای وابسته به ناظر ساده‌سازی کرد.

۸-۵ ساده‌سازی چندضلعی قابل دید ناظر پاره‌خطی

دربخش‌های قبلی، معیار خطای وابسته به ناظر را فقط برای ناظر نقطه‌ای تعریف کردیم. این تعریف را می‌توان برای گونه‌های دیگر ناظر تعمیم داد. فرض کنید که یک ناظر پاره‌خطی داریم و می‌خواهیم که چندضلعی قابل دید ضعیف و قوی آن را ساده کنیم.

۸-۵-۱ ساده‌سازی چندضلعی قابل دید ضعیف

فرض کنید $\mathcal{P}(i,j) = \langle p_i, p_{i+۱}, \ldots, p_j \rangle$ بخشی از چندضلعی قابل دید ضعیف، $WV(qr)$، مربوط به ناظر پاره‌خطی qr است که می‌خواهیم آن‌را با پاره‌خط p_ip_j ساده کنیم. اگر p_m یک نقطه از $\mathcal{P}(i,j)$ باشد، آنگاه بدیهی است که p_m از مجموعه‌ای از نقاط qr که با Q نشان می‌دهیم قابل دید است. اگر هر نقطه $q' \in Q$ را به عنوان یک ناظر نقطه‌ای در نظر بگیریم، آنگاه خطای فاصله‌ای وابسته به ناظر این ساده‌سازی برای نقطه p_m برابر است با $err_{vis}(p_m, p_ip_j)$. می‌دانیم که p_m در چندضلعی قابل

دید ضعیف qr قرار می‌گیرد حتی اگر فقط از یک نقطه از qr قابل دید باشد. بنابراین، طبیعی است که خطای فاصله‌ای وابسته به ناظر qr را برای نقطه p_m بصورت کمینه مقدار $err_{vis(q')}(p_m, p_ip_j)$، $q' \in Q$، تعریف کنیم. این خطا را با $err_{wvis(qr)}(p_m, p_ip_j)$ یا به صورت ساده‌تر با $err_{wvis}(p_m, p_ip_j)$ نشان می‌دهیم.

با داشتن مقدار خطای وابسته به دید نقاط $\mathcal{P}(i,j)$، خطای پاره‌خط p_ip_j را برابر با بیشینه مقادیر $err_{wvis}(p_m, p_ip_j)$ تعریف می‌کنیم که $p_m \in \mathcal{P}(i,j)$. این مقدار خطا را برای پاره‌خط p_ip_j با $err_{wvis(qr)}(p_ip_j)$ یا به صورت ساده‌تر با $err_{wvis}(p_ip_j)$ نشان می‌دهیم. بسادگی قابل اثبات است که این مقدار خطا مربوط به یک راس از $\mathcal{P}(i,j)$ است و در نتیجه برای محاسبه خطای هر پاره‌خط کافی است که خطای رئوس $\mathcal{P}(i,j)$ را محاسبه کنیم.

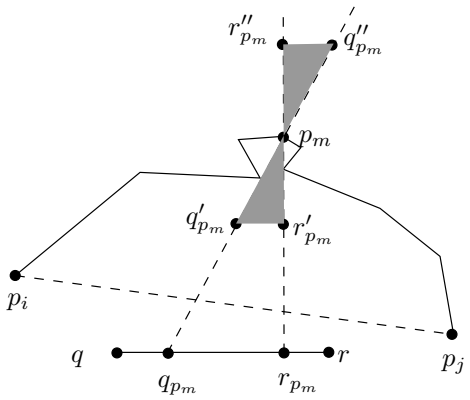
با این وجود، چندضلعی قابل دید یک ناظر پاره‌خطی واقع در یک دامنه چندضلعی‌گونه، یک چندضلعی ساده نیست. در واقع چندضلعی قابل دید ضعیف یک چندضلعی ساده است که درون آن تعدادی ناحیه محدب وجود دارد که از ناظر بصورت ضعیف قابل دید نیستند. بنابراین، اگر ساده‌سازی را روی مرز بیرونی یا مرز این ناحیه‌های محدب داخلی بصورت مستقل انجام دهیم، هیچ تضمینی وجود ندارد که یال‌های ساده‌سازی همدیگر را قطع نکنند. این یک مشکل اساسی در ساده‌سازی چندضلعی قابل دید ضعیف در دامنه‌های چندضلعی‌گونه است. در مقابل، اگر تنها مرز بیرونی را در نظر بگیریم این مساله وجود نخواهد داشت. اگر ناظر پاره‌خطی qr درون یک چندضلعی ساده باشد آنگاه $WV(qr)$ نیز یک چندضلعی ساده خواهد بود که برای آن می‌توانیم ساده‌سازی را بر اساس معیار تعریف شده بصورت کارایی انجام دهیم.

درون یک چندضلعی ساده، یک نقطه همواره از یک قطعه پیوسته از یک ناظر پاره‌خطی قابل دید است و طبق لم ۲۴، مقدار کمینه خطای وابسته به ناظر مربوط به یکی از دو انتهای این قطعه از ناظر است. این ویژگی باعث می‌شود که بتوانیم معیار ارائه شده برای ساده‌سازی چندضلعی قابل دید ضعیف را با استفاده از الگوریتم عمومی ارائه شده در بخش ۸—۲—۴ بکار ببریم.

برای این منظور، مرز $WV(qr)$ را پیمایش می‌کنیم تا برای هر راس آن قطعه قابل دید از ناظر را محاسبه کنیم. سپس، پاره‌خط‌هایی را که می‌توانند در مساله ساده‌سازی با کمترین راس ظاهر شوند باید مشخص شوند و گراف G_ϵ، همانند آنچه در مورد ناظر نقطه‌ای بیان کردیم، ساخته شود.

همانند ناظر نقطه‌ای، پاره‌خط p_ip_j در صورتی در ساده‌سازی با کمترین راس چندضلعی قابل دید ضعیف $WV(qr) = \mathcal{P}$ ظاهر می‌شود که برای همه نقاط p_m، $(i < m < j)$، رابطه $err_{wvis}(p_m, p_ip_j) \leq \epsilon$ برقرار باشد که ϵ مقدار خطای مجاز داده شده است. این بدان معنی است که p_ip_j باید حداقل یکی از پاره‌خط‌های $q'_{p_m}q''_{p_m}$ و $r'_{p_m}r''_{p_m}$ را که در شکل ۸—۷ نشان داده شده‌اند، قطع کند. در این شکل، $q_{p_m}r_{p_m}$ بخش قابل دید qr از نقطه p_m است و $q'_{p_m}q''_{p_m}$ و $r'_{p_m}r''_{p_m}$ به ترتیب ناحیه‌های نوسان نقطه p_m نسبت به ناظرهای نقطه‌ای q_{p_m} و r_{p_m} و خطای داده‌شده ϵ هستند.

بنابراین، $q'_{p_m}q''_{p_m}$ به G_ϵ اضافه می‌شود اگر و تنها اگر p_ip_j حداقل یکی از پاره‌خطهای $\overrightarrow{p_ip_j}$ و $r'_{p_m}r''_{p_m}$ را به ازای همه نقاط p_m، $(i < m < j)$، قطع کند. در نتیجه، کافی است که خط ۸ از رویه BuildDAG نشان داده شده در شکل ۸—۶ را با خط $PR_i = PR_i \cap (\angle q'_{p_j}p_iq''_{p_j} \cup \angle r'_{p_j}p_ir''_{p_j})$



شکل ۸–۷: ناحیه نوسان برای چندضلعی قابل دید ضعیف ناظر پاره‌خطی.

جایگزین کنیم تا گراف G_ϵ برای ساده‌سازی چندضلعی قابل دید ضعیف ساخته شود.

اندازه $WV(qr)$ در یک چندضلعی ساده n راسی از مرتبه $O(n)$ است و برای کلیه رئوس آن، پاره خط‌های $q_{p_m}r_{p_m}$ را می‌توان با یک الگوریتم بدیهی در زمان $O(n^2)$ محاسبه کرد. زمان اجرای رویه BuildDAG نیز $O(n^2)$ است. بنابراین، فرضیه زیر در دست است.

قضیه ۲۰ . چندضلعی قابل دید ضعیف یک ناظر پاره‌خطی واقع در یک چندضلعی ساده n راسی را می‌توان بر اساس معیار فاصله وابسته به ناظر در زمان $O(n^2)$ ساده کرد.

۸–۵–۲ ساده‌سازی چندضلعی قابل دید قوی

برخلاف چندضلعی قابل دید ضعیف، نقطه p_m در مرز $SV(qr)$ قرار می‌گیرد اگر از همه نقاط qr قابل دید باشد. بنابراین، خطای مربوط به نقطه p_m متعلق به مسیر $\mathcal{P}(i,j) = \langle p_i, p_{i+1}, \ldots, p_j \rangle$ از $SV(qr)$ را که بوسیله پاره‌خط p_ip_j ساده شده است برابر با بیشینه مقدار $err_{vis(q')}(p_m, p_ip_j)$ تعریف می‌کنیم که $q' \in qr$. این مقدار خطا را با $err_{svis(q')}(p_m, p_ip_j)$ یا به صورت ساده‌تر با $err_{svis}(p_m, p_ip_j)$ نشان می‌دهیم.

از لم ۲۴ نتیجه می‌گیریم که این مقدار بیشینه به ازای $q' = q$ یا $q' = r$ حاصل می‌شود. بنابراین، محاسبه این خطا به ازای هر نقطه از مسیر $\mathcal{P}(i,j)$ کاملا سرراست است.

مشابه ناظر نقطه‌ای یا چندضلعی قابل دید ضعیف، خطای فاصله‌ای وابسته به ناظر را برای پاره‌خط p_ip_j بصورت بیشینه مقدار $err_{svis(qr)}(p_m, p_ip_j)$ روی همه نقاط p_m از $\mathcal{P}(i,j)$ تعریف می‌کنیم و آن‌را با $err_{svis(qr)}(p_ip_j)$ یا به صورت ساده‌تر با $err_{svis}(p_ip_j)$ نشان می‌دهیم. در اینجا نیز به سادگی می‌توان نشان داد که $err_{svis}(p_ip_j)$ برابر است با $err_{svis(qr)}(p_m, p_ip_j)$ که p_m یکی از رئوس مسیر $\mathcal{P}(i,j)$ است. این مطلب محاسبه $err_{svis}(p_ip_j)$ را ساده می‌کند و کافی است که خطای

مربوط به هر راس از مسیر $\mathcal{P}(i,j)$ را حساب کنیم و بیشنیه آنها را به عنوان $err_{svis}(p_ip_j)$ در نظر بگیریم.

این معیار را می‌توان برای ساده‌سازی $SV(qr)$ با استفاده از الگوریتم عمومی ارائه شده در بخش ۸–۲–۴ بکار برد. برای این کار کافی است که گراف G_ϵ مورد استفاده در الگوریتم عمومی را بر اساس این معیار بسازیم. فرض کنید پاره‌خط‌های $q'_{p_m}q''_{p_m}$ و $r'_{p_m}r''_{p_m}$ به ترتیب ناحیه‌های نوسان نقطه p_m نسبت به ناظرهای نقطه‌ای q و r و خطای مجاز ϵ هستند. در این صورت، یال $\overrightarrow{p_ip_j}$ به گراف G_ϵ اضافه می‌شود اگر و تنها اگر هر دو پاره‌خط p_ip_j و $q'_{p_m}q''_{p_m}$ و $r'_{p_m}r''_{p_m}$ را به ازای همه نقاط p_m، $(i<m<j)$، قطع کند. بنابراین، کافی است که خط ۸ از رویه BuildDAG نشان داده شده در شکل ۸–۶ را با خط $PR_i = PR_i \cap (\angle q'_{p_j}p_iq''_{p_j} \cap \angle r'_{p_j}p_ir''_{p_j})$ جایگزین کنیم. بدین ترتیب گراف مورد نیاز الگوریتم عمومی برای ساده‌سازی با کمترین راس بر اساس معیار فاصله‌ای وابسته به ناظر بر روی چندضلعی قابل دید قوی یک ناظر پاره‌خطی در زمان $O(n^2)$ بدست می‌آید. اندازه چندضلعی قابل دید قوی در دامنه‌های چندضلعی‌گونه $O(n)$ است [۱۲۴، ۷۱]. در نتیجه، قضیه زیر را داریم.

قضیه ۲۱ . چندضلعی قابل دید قوی یک ناظر پاره‌خطی واقع در یک دامنه چندضلعی‌گونه n راسی را می‌توان بر اساس معیار فاصله وابسته به ناظر در زمان $O(n^2)$ ساده‌سازی کرد.

۸–۶ الگوریتم کارای ساده‌سازی چندضلعی قابل دید ناظر پاره‌خطی

در بخش ۸–۴ یک الگوریتم کارا با مرتبه زمانی و حافظه مصرفی $O(n^{4/3+\delta})$ برای ساده‌سازی چندضلعی قابل دید ناظر نقطه‌ای ارائه شد که مبتنی بر نگهداری فشرده گراف G_ϵ مربوط به الگوریتم عمومی ساده‌سازی است. در این بخش، این روش را برای ساده‌سازی چندضلعی قابل دید ناظر پاره‌خطی بکار می‌بریم.

برای استفاده از این روش، کافی است که $cone(p_i)$ را متناسب با ناحیه‌های نوسان جدید محاسبه کنیم. بقیه اجزای الگوریتم و استدلال‌های انجام شده بدون تغییر قابل استفاده‌اند. در مورد چندضلعی قابل دید قوی، هر راس p_i دارای دو پاره‌خط $q'_{p_i}q''_{p_i}$ و $r'_{p_i}r''_{p_i}$ است که $cone(p_i)$ باید درون این دو پاره‌خط قرار گیرد. برای این منظور، هنگام محاسبه $cone(p_i)$ طبق لم ۲۸، L_i براساس نقاط $\{q'_{p_i}, r'_{p_i}, q'_{p_{i+1}}, r'_{p_{i+1}}, \ldots, q'_{p_{\lfloor \frac{n}{2} \rfloor}}, r'_{p_{\lfloor \frac{n}{2} \rfloor}}\}$ و U_i براساس نقاط $\{q''_{p_i}, r''_{p_i}, q''_{p_{i+1}}, r''_{p_{i+1}}, \ldots, q''_{p_{\lfloor \frac{n}{2} \rfloor}}, r''_{p_{\lfloor \frac{n}{2} \rfloor}}\}$ محاسبه می‌شود. بقیه روش و استدلال ارائه شده در لم ۲۸ بدون تغییر قابل استفاده است. در نتیجه، قضیه زیر در دست است.

قضیه ۲۲ . چندضلعی قابل دید قوی یک ناظر پاره‌خطی واقع در یک دامنه چندضلعی‌گونه را می‌توان با صرف زمان و حافظه $O(n^{4/3+\delta})$ بر اساس معیار فاصله‌ای وابسته به ناظر ساده‌سازی کرد.

در مورد چندضلعی قابل دید ضعیف یک ناظر پاره‌خطی واقع در یک چندضلعی ساده نیز کافی است $cone(p_i)$ را مطابق با نسخه اصلاح شده لم ۲۸ بدست آوریم. بقیه استدلال‌ها و روش‌ها بدون تغییر قابل استفاده هستند. برخلاف چندضلعی قابل دید قوی، در اینجا هر شعاع موجود در $cone(p_i)$ کافی است که یکی از پاره‌خط‌های $q'_{p_i}q''_{p_i}$ یا $r'_{p_i}r''_{p_i}$ را قطع کند. در نتیجه، استفاده از L_i و U_i به روش بالا در اینجا کافی نیست. برای حل این مشکل به این ترتیب عمل می‌کنیم که هنگام بهنگام‌سازی L_i از بین دو راس q'_{p_i} و r'_{p_i}، راسی در نظر گرفته می‌شود که به ازای آن راس پوش محدب نقاط L_i کوچک‌تر باشد. به عبارت دیگر، اگر با افزودن یکی از این راس‌ها، مثلا q'_{p_i}، به L_i راس دیگر یعنی r'_{p_i} درون پوش محدب نقاط قرار گیرد آنگاه q'_{p_i} به L_i اضافه نمی‌شود و به جای آن r'_{p_i} به L_i اضافه می‌شود. همچنین، اگر با افزودن راس r'_{p_i} به L_i، راس q'_{p_i} درون پوش محدب L_i قرار گیرد آنگاه بجای r'_{p_i} راس q'_{p_i} به L_i اضافه می‌شود. در مورد U_i نیز به همین ترتیب از بین دو راس q''_{p_i} و r''_{p_i} فقط راسی اضافه می‌شود که به ازای آن پوش محدب نقاط U_i کوچک‌تر باشد. اعمال این تغییرات در محاسبه $cone(p_i)$ تغییری در زمان اجرای لم ۲۸ ایجاد نمی‌کند. درنتیجه، قضیه زیر را داریم.

قضیه ۲۳ . چندضلعی قابل دید ضعیف یک ناظر پاره‌خطی واقع در یک چندضلعی ساده را می‌توان با صرف زمان و حافظه $O(n^{۴/۳+\delta})$ بر اساس معیار فاصله‌ای وابسته به ناظر ساده‌سازی کرد.

۸-۷ نتیجه‌گیری

در این فصل، ساده‌سازی مرز ناحیه قابل دید یک ناظر بررسی شد. این مساله در کاربردهای واقعی که حافظه موجود محدود و تعداد نقاط روی مرز ناحیه قابل دید بسیار زیاد است مفید است. همچنین، در مواردی که نگهداری دقیق این مرز به علت محدودیت دقت صفحات نمایش مفید نیست، ساده‌سازی مرز ناحیه قابل دید باعث افزایش کارایی الگوریتم‌های مربوط می‌شود.

با توجه به اینکه معیارهای ساده‌سازی موجود برای حل این مساله مناسب نیستند، ابتدا دو معیار اصلی ساده‌سازی را برای این مساله خاص بازتعریف کردیم. برای معیارهای جدید، نشان دادیم که چگونه می‌توان آنها را به همراه الگوریتم‌های ساده‌سازی موجود برای ساده‌سازی مرز ناحیه‌های قابل دید از ناظر نقطه‌ای یا پاره‌خطی بکار برد.

مساله‌ای که در این فصل بررسی شد و نتایجی که به‌دست آمد نخستین نتایج در این حوزه کاربردی از مسائل هندسه محاسباتی است و در نتیجه زمینه‌های متعددی برای توسعه و ادامه آنها وجود دارد.

معیارهای ساده‌سازی ارائه شده برای ناظرهای نقطه‌ای و پاره‌خطی به گونه‌های دیگر ناظرها از جمله ناظرهای چندضلعی‌گونه نیز قابل توسعه هستند. همچنین، بکارگیری این معیارها و

ساده‌سازی مرز ناحیه قابل دید از یک ناظر متحرک مساله جالب دیگری است که در ادامه این کار قابل انجام است. نکته قابل توجه برای این مساله این است که در اینجا ساده‌سازی باید بصورت برخط انجام شود.

ساده‌سازی ناحیه قابل دید در فضای سه‌بعدی مساله باز و کاربردی دیگری است که تعمیم و بکارگیری روش‌های بدست آمده در این فصل برای آن قابل مطالعه و بررسی است.

همچنین، پیاده‌سازی و ارزیابی عملی روش‌ها و نتایج بدست آمده در این فصل برای مقایسه کیفیت این روش‌ها نسبت به روش‌های موجود ارزشمند خواهد بود.

فصل ۹

ساده‌سازی در مدل جویباری

در فصل ۸، ساده‌سازی چندضلعی قابل دید را برای حالتی که کل چندضلعی در دسترس بود و امکان پردازش بر روی کل آن وجود داشت بررسی کردیم. در این فصل، حالتی از مساله ساده‌سازی را بررسی می‌کنیم که رئوس مسیر به صورت یک دنباله نامتناهی و بصورت جویباری داده می‌شوند. این دنباله از نقاط $p_0, p_1, p_2, \ldots$ در صفحه یک مسیر تشکیل می‌دهند که با رسیدن هر راس جدید می‌خواهیم دنباله ساده شده بخشی از مسیر را که تاکنون دریافت کرده‌ایم، بسازیم.

در این مدل که به آن مدل ورودی جویباری[۱] گفته می‌شود، امکان نگهداری کلیه نقاط مسیر و تهیه مسیر ساده شده بهینه طبق الگوریتم‌های ارائه شده در فصل ۸ وجود ندارد. برای این منظور، یک الگوریتم ساده‌سازی جدید ارائه می‌دهیم که مسیر ساده شده را بصورت تقریبی برای مساله کمترین خطا محاسبه می‌کند. ضریب رقابتی[۲] این الگوریتم را با درنظر گرفتن هم‌افزایی منابع[۳] تحلیل می‌کنیم.

در الگوریتم ارائه شده حداکثر $2k$ راس میانی ظاهر می‌شود و خطای آن‌را با خطای ساده‌سازی بهینه‌ای که حداکثر k راس میانی دارد مقایسه می‌کنیم. ضریب رقابتی این الگوریتم برای ساده‌سازی هر مسیر دلخواه محدب بر اساس معیار فاصله هاوس‌دورف، هر مسیر xy-یکنوا بر اساس معیار فاصله هاوس‌دورف، هر مسیر دلخواه براساس معیار فاصله فِرشه، چندضلعی قابل دید ناظر نقطه‌ای براساس معیار فاصله‌ای وابسته به ناظر و چندضلعی قابل دید قوی ناظر پاره‌خطی بر اساس معیار فاصله‌ای وابسته به ناظر از مرتبه $O(1)$ است. حافظه مورد نیاز این الگوریتم برای مورد اول $O(k)$ و برای چهارمورد بعدی $O(k^2)$ است.

---

[۱] Streaming Input Model

[۲] Compatitive Ratio

[۳] Resource Augmentation

۹-۱ مقدمه

فرض کنید که می‌خواهیم موقعیت و مسیر حرکت یک شی را دنبال و نگهداری کنیم. به این شی یک وسیله متصل شده است که بصورت پیوسته موقعیت آن را ارسال می‌کند. بنابراین جویباری از اطلاعات نقاط را دریافت می‌کنیم که توصیف کننده مسیری است که شی پیموده است و هدف نگهداری این مسیر به ازای این گونه اشیاء است. در برخی موارد، از جمله تشخیص الگوهای حرکتی جانوران مهاجر، امکان نگهداری کل این داده‌های جویباری وجود ندارد یا نگهداری همه آنها غیر ضروری است. در مقابل، کافی است که تقریب ساده شده‌ای از این مسیر نگهداری شود که منجر به مساله زیر می‌شود: یک جویبار (احتمالا نامتناهی) از نقاط $p_0, p_1, \ldots$ دریافت می‌شود و انتظار داریم که ساده شده مسیری را که تاکنون دریافت کرده‌ایم نگهداری کنیم بطوریکه تا جای ممکن شبیه به مسیر اولیه باشد و تعداد نقاط مسیر ساده شده نیز از مقدار ثابت حافظه موجود بیشتر نباشد.

این مساله در کاربردهای قابلیت دید نیز وجود دارد: ناظری را در نظر بگیرید که درون یک ناحیه مسطح قرار دارد. این ناظر بصورت دایره‌ای ناحیه اطراف خود را جاروب می‌کند و مرز ناحیه قابل دید را بصورت پیوسته می‌کشد. در این گونه کاربردها نیز نقاط قابل دید بصورت جویباری از نقاط دریافت می‌شوند و با فرض نامتناهی بودن این تعداد از نقاط، امکان نگهداری آنها وجود نخواهد داشت. بنابراین، باید تقریبی از مرز این ناحیه دید را با استفاده از تعداد راس‌های کمتری ثبت و نگهداری کرد.

این مساله نسخه جویباری مساله ساده‌سازی است که در فصل ۸ به آن پرداخته شد. یادآوری می‌کنیم که در مساله ساده‌سازی، مسیر چندضلعی‌گونه $\mathcal{P} = \langle p_0, p_1, \ldots, p_n \rangle$ در صفحه موجود است و هدف یافتن مسیر $\mathcal{Q} = \langle q_0, q_1, \ldots, q_{k+1} \rangle$ با تعداد کمتری راس است که تقریب مناسبی از $\mathcal{P}$ را ارائه می‌دهد. این مساله هر جا که کاهش داده‌ها در شکل‌های چندضلعی‌گونه لازم باشد مطرح می‌شود و نقش اساسی در GIS، پردازش تصویر و گرافیک کامپیوتری دارد.

مساله ساده‌سازی مسیر بصورت عمیق در این حوزه‌ها و نیز در هندسه محاسباتی مطالعه شده است. ما در این فصل حالت جویباری مساله ساده‌سازی را بررسی می‌کنیم که تاکنون مورد مطالعه قرار نگرفته است. نمونه‌های دیگری از مساله جویباری در هندسه محاسباتی نیز بررسی شده‌است و مسائل دیگری از هندسه محاسباتی در مدل جویباری حل شده‌است که مشابه با مدل استفاده شده در این فصل است [۸، ۹۸، ۱۴۸]. قبل از پرداختن به مساله ساده‌سازی در مدل جویباری، نتایجی را که برای این مساله در حالت غیرجویباری بدست آمده است بیان می‌کنیم.

مساله ساده‌سازی مسیر دارای گونه‌های مختلفی است که ساده‌سازی مقید و بی‌قید دو گونه مهم آن است. در این فصل فقط گونه مقید آن را بررسی می‌کنیم. برخی از نتایج مربوط به نسخه بی‌قید آن در مراجع [۵۸، ۶۳، ۶۶، ۷۹] آمده‌است. در نسخه مقید ساده‌سازی، هر پاره‌خط $q_l q_{l+1}$ از مسیر ساده شده متناظر با پاره‌خط $p_i p_j$، $(i < j)$، از مسیر اصلی است و خطای متناظر با این پاره‌خط بر حسب فاصله بین $q_l q_{l+1}$ و مسیر $\mathcal{P}(i, j)$ بدست می‌آید. برای اندازه‌گیری این فاصله معمولا

از فاصله هاوس‌دورف استفاده می‌شود. سپس، خطای ساده‌سازی $\mathcal{Q}$ را برابر با بیشینه خطای پاره‌خط‌های q_iq_{i+1} آن تعریف می‌کنیم. با داشتن تعریف معیار خطا، دو مساله بهینه‌سازی ساده‌سازی با کمترین راس و ساده‌سازی با کمترین خطا بررسی شده‌اند. در مساله ساده‌سازی با کمترین راس، هدف یافتن ساده‌سازی $\mathcal{Q}$ با کمترین تعداد راس است که خطای آن از مقدار داده شده δ بیشتر نباشد. در مساله ساده‌سازی با کمترین خطا، هدف یافتن ساده‌سازی $\mathcal{Q}$ با کمترین خطا است که تعداد راس‌های آن بیشتر از مقدار داده شده k نباشد.

الگوریتم Douglas-Peucker قدیمی‌ترین و معروف‌ترین الگوریتم ساده‌سازی برحسب معیار خطای فاصله هاوس‌دورف است [۳۷]. زمان اجرای پیاده‌سازی ابتدایی این الگوریتم $O(n^2)$ است ولی پیاده‌سازی مناسب‌تر آن دارای زمان اجرای $O(n\log n)$ [۷۳] و $O(n\log^* n)$ [۷۴] است. به هر حال، این الگوریتم یک الگوریتم ابتکاری است که هیچ تضمینی در مورد بهینه بودن جواب بر حسب تعداد رئوس یا خطای آن ارائه نمی‌دهد. Imai و Iri [۷۴] این مساله را بصورت بهینه (از نظر خطای ساده سازی) در زمان $O(n^2\log n)$ حل کردند. آنها مساله ساده‌سازی را با یک گراف جهت‌دار مدل‌سازی کردند و با حل مساله کوتاه‌ترین مسیر در این گراف آن را حل کردند. زمان اجرای الگوریتم آنها سپس به $O(n^2)$ بهبود داده شد [۲۸، ۹۶]. درنهایت، زمان اجرای این الگوریتم با نگهداری غیرصریح گراف جهت‌دار به $O(n^{4/3+\epsilon})$، برای معیار خطای L_1[۴] و یکنواخت[۵]، برای مقدار دلخواه $\epsilon > 0$ کاهش یافت [۹۶].

مساله ساده‌سازی بر اساس معیار خطای فِرشه نخستین بار توسط Godau بررسی شد [۵۷]. Godau و Alt روشی ارائه دادند که می‌تواند مساله ساده‌سازی با کمترین راس و کمترین خطا را بر اساس معیار خطای فاصله فِرشه حل کند. از آنجا که زمان اجرای حل دقیق مساله ساده‌سازی بالا است (زمان اجرای بهترین الگوریتم ساده‌سازی بر اساس فاصله هاوس‌دورف و فِرشه $O(n^2)$ و بالاتر است.)، حل تقریبی این مساله نیز بررسی شده است. در همین راستا، الگوریتمی تقریبی برای حل مساله ساده‌سازی با کمترین راس برای معیار خطای فاصله هاوس‌دورف برای مسیرهای x-یکنوا در صفحه و نیز معیار خطای فاصله فِرشه برای مسیرهای عمومی در فضای d-بعدی ارائه شده‌است [۵]. زمان اجرای این الگوریتم تقریبا خطی است و خطای ساده‌سازی حاصل از آن برابر δ است که تعداد راس‌های موجود در این ساده‌سازی حداکثر برابر با تعداد رئوس ساده‌سازی بهینه با خطای $\delta/2$ است. این الگوریتم حریصانه و مرحله‌ای است. با توجه به مرحله‌ای بودن این الگوریتم، آن‌را می‌توان در حالت‌های برخط که نقاط مسیر اصلی بصورت متوالی داده می‌شوند و ساده‌سازی را باید در هر مرحله بهنگام کرد، بکار برد. با این وجود، این الگوریتم مساله ساده‌سازی با کمترین راس را حل می‌کند و با توجه به اینکه در این مساله تعداد رئوس مسیر ساده‌شده ممکن است $\theta(n)$ باشد، نمی‌توان آن‌را برای حالت‌های جویباری استفاده کرد. این الگوریتم حریصانه و مرحله‌ای برای حل مساله کمترین راس به این ترتیب عمل می‌کند که هر پاره‌خط را تا زمانی که

[۴] در معیار L_1، فاصله دو نقطه (x_1,y_1) و (x_2,y_2) برابر است با $(x_1-x_2)+(y_1-y_2)$.

[۵] در معیار یکنوا، فاصله دو نقطه (x_1,y_1) و (x_2,y_2) برابر است با $|y_1-y_2|$ اگر $x_1=x_2$ و در غیر این صورت برابر است با بینهایت.

خطای آن از مقدار مجاز بیشتر نشده است در ساده‌سازی نگه می‌دارد ولی این ایده برای حل مساله ساده‌سازی با کمترین خطا بکار نمی آید. علاوه‌برآن، این الگوریتم تقریبی برای معیار خطای فاصله هاوس‌دورف فقط برای فاصله یکنوا کار می‌کند و اگر فاصله $L_۲$ را در نظر بگیریم نمی‌توان از آن استفاده کرد. سایر الگوریتم‌های ساده‌سازی را نیز نمی‌توان برای حالت جویباری استفاده کرد.

در ادامه این فصل، تعاریف و نمادهای مورد استفاده و نتایج بدست آمده در بخش ۹—۲ بیان می‌شود. در بخش ۹—۳ الگوریتم عمومی ساده‌سازی در حالت جویباری ارائه می‌شود. در بخش‌های ۹—۴، ۹—۵ و ۹—۶ به ترتیب ساده‌سازی بر اساس معیار خطای فاصله هاوس‌دورف، ساده‌سازی بر اساس معیار خطای فاصله فِرشه و ساده‌سازی بر اساس معیار خطای فاصله‌ای وابسته به ناظر بیان می‌شود.

۹-۲ تعریف مساله و نتایج بدست آمده

فرض کنید جویبار نقاط ورودی شامل $p_۰, p_۱, p_۲, \ldots$ است و $\mathcal{P}(n)$ بیانگر مسیر تشکیل شده از نقاط $p_۰, p_۱, \ldots, p_n$ است و به ازای هر دو نقطه p_i و p_j از این مسیر $\mathcal{P}(p_i, p_j)$ نشان دهنده بخشی از مسیر $\mathcal{P}(n)$ است که شامل نقاط p_i تا p_j است که به صورت ساده‌تر با $\mathcal{P}(i, j)$ نیز نشان داده می‌شود. پاره‌خط p_ip_j، $i < j$، یک اتصال یا میانبر نامیده می‌شود که می‌تواند در ساده‌سازی $\mathcal{P}(n)$ ظاهر شود. $\mathcal{P}(n)$ شامل تمام اتصالات $p_ip_{i+۱}$ به ازای $۰ \leq i < n$ است.

فرض کنید تابع err مقدار غیر منفی خطای هر اتصال را مشخص می‌کند. دنباله $\mathcal{Q} = q_۰, q_۱, \ldots, q_k, q_{k+۱}$ برای $k \leq l$ یک l–ساده‌سازی از $\mathcal{P}(n)$ است اگر $q_۰ = p_۰$، $q_{k+۱} = p_n$ و $q_۱, \ldots, q_k$ یک زیردنباله از $p_۱, p_۲, \ldots, p_{n-۱}$ باشد. خطای ساده‌سازی $\mathcal{Q}$ براساس تابع خطای err را با $err(\mathcal{Q})$ نشان می‌دهیم و برابر با بیشینه خطای میانبرهای آن تعریف می‌کنیم. در این فصل سه معیار خطای فاصله هاوس‌دورف، فاصله فِرشه و فاصله وابسته به ناظر را به عنوان تابع خطا استفاده می‌کنیم. معیار خطای فاصله‌ای وابسته به ناظر را در فصل ۸ برای ناظر نقطه‌ای و پاره‌خطی تعریف کردیم. دو معیار خطای فاصله هاوس‌دورف و فاصله فِرشه در این بخش تعریف می‌شوند. فاصله اقلیدسی دو شی $o_۱$ و $o_۲$ را با $d(o_۱, o_۲)$ نشان می‌دهیم. اگر $o_۱$ و $o_۲$ دو نقطه باشند آنگاه این فاصله بصورت $|o_۱o_۲|$ نیز نشان داده می‌شود که برابر با فاصله بین این دو نقطه است.

تابع خطای مربوط به فاصله هاوس‌دورف را با err_H نشان می‌دهیم. خطای اتصال p_ip_j در این معیار با $err_H(p_ip_j)$ نشان داده می‌شود و مقدار آن برابر با فاصله هاوس‌دورف بین $\mathcal{P}(i, j)$ و p_ip_j است که با $d_H(p_ip_j, \mathcal{P})$ نشان داده می‌شود و مقدار آن برابر با $\max_{i<l<j} d(p_l, p_ip_j)$ تعریف می‌شود.

فاصله فِرشه بین دو مسیر A و B را با $d_F(A, B)$ نشان می‌دهیم که بصورت زیر تعریف می‌شود. یک آدم و یک سگ را در نظر بگیرید که با هم پیاده‌روی می‌کنند و موقعیت اولیه آدم در ابتدای مسیر A و موقعیت اولیه سگ در ابتدای مسیر B است. آدم مسیر A را می‌پیماید و سگ در

مسیر B حرکت می‌کند و در خلال این حرکت می‌توانند به جلو حرکت کنند یا در محل خود بایستند ولی نمی‌توانند در مسیر خود به عقب حرکت کنند. فاصله فِرشه بین دو مسیر A و B برابر با کوتاه‌ترین طول برای نگهداری سگ توسط آدم روی تمام حالات ممکن برای انجام این پیاده‌روی تعریف می‌شود. بصورت رسمی، $d_F(A,B)$ بدین صورت تعریف می‌شود: فرض کنید مسیرهای A و B با توابع $A : [۰,۱] \to R^۲$ و $B : [۰,۱] \to R^۲$ تعریف شده‌اند. هر تابع پیوسته و ناکاهشی $\alpha : [۰,۱] \to [۰,۱]$ که $\alpha(۰) = ۰$ و $\alpha(۱) = ۱$ یک پارامتری کردن جدید A_α برای A است که در آن $A_\alpha(t) = A(\alpha(t))$. بصورت مشابه، هر تابع پیوسته و ناکاهشی $\beta : [۰,۱] \to [۰,۱]$ که $\beta(۰) = ۰$ و $\beta(۱) = ۱$ یک پارامتری کردن جدید B_β برای B است که در آن $B_\beta(t) = B(\beta(t))$. فاصله فِرشه بین دو مسیر A و B بصورت

$$d_F(A,B) = \inf_{\alpha,\beta} \max_{۰\le t\le ۱} d(A_\alpha(t), B_\beta(t))$$

تعریف می‌شود که inf روی همه پارامتری کردن‌های A_α برای A و B_β برای B انجام می‌شود. در معیار فِرشه، تابع خطا را با err_F نشان می‌دهیم و خطای میان‌بر p_ip_j برحسب این معیار خطا با $err_F(p_ip_j)$ نشان داده می‌شود و مقدار آن برابر است با $d_F(p_ip_j, \mathcal{P}(i,j))$.

حال الگوریتم $A = A(l)$ را در نظر بگیرید که یک l–ساده‌سازی از مسیر جویباری $p_۰, p_۱, \ldots$ را برای مقدار داده شده l محاسبه می‌کند. همچنین فرض کنید $\mathcal{Q}_A(n)$ ساده‌سازی حاصل از A برای مسیر $\mathcal{P}(n)$ است و $Opt(l)$ یک الگوریتم غیرجویباری بهینه برای l–ساده‌سازی است. بنابراین، $err(\mathcal{Q}_{Opt(l)}(n))$ کمترین مقدار خطای ممکن برای همه l–ساده‌سازی‌های $\mathcal{P}(n)$ است.

ما کیفیت الگوریتم A را با استفاده از یک ضریب رقابتی ارزیابی می‌کنیم که روشی استاندارد برای الگوریتم‌های برخط است. در این ارزیابی امکان هم‌افزایی منابع نیز در نظر گرفته می‌شود. بطور دقیق‌تر، در الگوریتم A ساده‌سازی با $۲k$ راس میانی انجام می‌شود ولی خطای ساده‌سازی بدست آمده با $\mathcal{Q}_{Opt(k)}(n)$ مقایسه می‌شود. این مشابه روش ارائه شده در [۵] است که جواب بدست آمده برای مساله ساده‌سازی با کمترین راس و مقدار خطای داده شده δ را با جواب بهینه خطای داده شده $\delta/۲$ مقایسه می‌کند.

ضریب رقابتی الگوریتم $A(۲k)$ را بصورت

$$A(۲k) \text{ ضریب رقابتی} = \max_{n\ge ۰} \frac{err(\mathcal{Q}_{A(۲k)}(n))}{err(\mathcal{Q}_{Opt(k)}(n))}$$

تعریف می‌کنیم که $\frac{err(\mathcal{Q}_{A(۲k)}(n))}{err(\mathcal{Q}_{Opt(k)}(n))}$ برابر با ۱ تعریف می‌شود اگر $err(\mathcal{Q}_{A(۲k)}(n)) = err(\mathcal{Q}_{Opt(k)}(n)) = ۰$. اگر ضریب رقابتی یک الگوریتم حداکثر c باشد آن را یک الگوریتم c–رقابتی می‌نامیم.

۹-۲-۱ نتایج بدست آمده

ما ابتدا یک الگوریتم عمومی برای ساده‌سازی در مدل ورودی جویباری ارائه می‌دهیم. نتایج تحلیل این الگوریتم نشان می‌دهد که با داشتن دو شرط، ضریب رقابتی این الگوریتم خوب خواهد بود: تابع خطای err استفاده شده، یکنوا باشد (مطابق با تعریف ارائه شده در بخش ۹—۳) و روشی برای تخمین خطای میان‌برهایی که در الگوریتم مورد استفاده قرار می‌گیرند وجود داشته باشد.

نشان می‌دهیم که تابع خطای فاصله هاوس‌دورف بر روی مسیرهای محدب و xy-یکنوا یکنوا است ولی برروی مسیرهای عمومی یکنوا نیست؛ تابع خطای فاصله فِرشه بر روی هر مسیر دلخواهی یکنوا است؛ تابع خطای فاصله وابسته به ناظر برای چندضلعی قابل دید ناظر نقطه‌ای و چندضلعی قابل دید قوی ناظر پاره‌خطی یکنوا است. همچنین برای هرکدام از موارد بالا روشی برای تخمین خطا ارائه می‌دهیم. در مجموع نتایج زیر بدست آمده است:

۱) برای مسیرهای محدب و معیار خطای هاوس‌دورف یا فِرشه یک الگوریتم ۳-رقابتی بدست می‌آوریم که حافظه مصرفی آن $O(k)$ است و هر نقطه ورودی جدید در زمان $O(\log k)$ پردازش می‌شود.

۲) برای مسیرهای xy-یکنوا و معیار خطای فاصله هاوس‌دورف یا فِرشه به ازای هر مقدار ثابت $\epsilon > 0$ یک الگوریتم جویباری $(4+\epsilon)$-رقابتی ارائه می‌کنیم که حافظه مورد نیاز آن $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ است و هر نقطه ورودی جدید در زمان سرشکنی $O(k \log \frac{1}{\epsilon})$ پردازش می‌شود.

۳) برای یک مسیر دلخواه و معیار خطای فاصله فِرشه به ازای هر مقدار ثابت $\epsilon > 0$ یک الگوریتم جویباری $(4\sqrt{2}+\epsilon)$-رقابتی ارائه می‌کنیم که حافظه مورد نیاز آن $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ است و هر نقطه ورودی جدید در زمان سرشکنی $O(k \log \frac{1}{\epsilon})$ پردازش می‌شود.

۴) برای چندضلعی قابل دید یک ناظر نقطه‌ای و نیز چندضلعی قابل دید قوی یک ناظر پاره‌خطی در دامنه‌های چندضلعی‌گونه و معیار خطای فاصله وابسته به ناظر به ازای هر مقدار ثابت $\epsilon > 0$ یک الگوریتم جویباری $(2+\epsilon)$-رقابتی ارائه می‌کنیم که حافظه مورد نیاز آن $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ است و هر نقطه ورودی جدید در زمان سرشکنی $O(k \log \frac{1}{\epsilon})$ پردازش می‌شود.

۵) نشان می‌دهیم که برای معیار خطای فاصله هاوس‌دورف هیچ الگوریتم جویباری با $2k$ نقطه میانی و ضریب رقابتی محدود (کوچکتر از یک مقدار ثابت) نسبت به $Opt(k)$ برای هر مسیر دلخواه وجود ندارد مگر اینکه حافظه مصرفی آن $\Omega(n/k)$ باشد.

۹-۳ الگوریتم عمومی ساده‌سازی جویباری

در این بخش، یک الگوریتم عمومی برای نگهداری یک l–ساده‌سازی از نقاط $p_0, p_1, \ldots$ در صفحه که به صورت جویباری داده می‌شوند ارائه می‌کنیم. نشان می‌دهیم که ضریب رقابتی این الگوریتم با داشتن دو شرط مناسب است: تابع خطای استفاده شده در ساده‌سازی بر اساس تعریف زیر یکنوا باشد و رویه‌ای برای محاسبه تخمین خطای میان‌برهایی که در الگوریتم استفاده می‌شوند وجود داشته باشد. خطای محاسبه‌شده توسط این رویه را برای میان‌بر p_ip_j و تابع خطای err با $err^*(p_ip_j)$ نشان می‌دهیم. در بخش‌های بعدی نشان می‌دهیم که این دو شرط برای پنج حالت از زوجِ مسیر و معیار خطا برقرار است و در نتیجه این الگوریتم برای آنها قابل استفاده است.

الگوریتم مورد نظر کاملا ساده است. فرض کنید که ما نقاط $p_0, p_1, \ldots, p_n$ را پردازش کرده‌ایم و $\mathcal{Q} = q_0, q_1, \ldots, q_l, q_{l+1}$ ساده‌سازی بدست آمده است. همچنین فرض می‌کنیم که $n > l + 1$ زیرا تا قبل از این مقطع همه نقاط را می‌توانیم در ساده‌سازی بکار ببریم و یک l–ساده‌سازی با خطای صفر داشته باشیم. در این الگوریتم یک صف اولویت $\mathcal{T}$ وجود دارد که در آن رئوس q_i به ازای $1 \leq i \leq l$ نگهداری می‌شوند و اولویت هر راس q_i برابر با خطای محاسبه شده توسط رویه تخمین خطا برای میان‌بر $q_{i-1}q_{i+1}$ یعنی $err^*(q_{i-1}q_{i+1})$ است. به عبارت دیگر، اولویت راس q_i برابر با (تخمینی از) خطایی است که از حذف راس q_i از ساده‌سازی بوجود می‌آید. حال راس بعدی p_{n+1} بصورت زیر پردازش می‌شود.

۱) q_{l+2} را برابر با p_{n+1} می‌گیریم که در نتیجه آن یک $(l+1)$–ساده‌سازی برای $\mathcal{P}(n+1)$ بدست می‌آید.

۲) خطای میان‌بر q_lq_{l+2} محاسبه می‌شود و q_{l+2} با اولویت $err^*(q_lq_{l+2})$ در $\mathcal{T}$ درج می‌شود.

۳) راس p_s که کمترین اولویت را دارد از $\mathcal{T}$ و نیز از ساده‌سازی حذف می‌شود.

۴) اولویت‌های رئوس q_{s-1} و q_{s+1} مجددا محاسبه و براساس این مقادیر جدید $\mathcal{T}$ بهنگام‌سازی می‌شود.

بدیهی است که این الگوریتم پس از پردازش p_{n+1} یک l–ساده‌سازی برای $\mathcal{P}(n+1)$ تولید می‌کند. در ادامه ضریب رقابتی این الگوریتم را محاسبه و آن را تحلیل می‌کنیم.

می‌گوییم میان‌بر p_ip_j میان‌بر p_lp_m را دربر می‌گیرد هر گاه $i \leq l \leq m \leq j$. برای مسیر $\mathcal{P}(n)$ تابع خطای err یکنوا است اگر ثابت c وجود داشته باشد بطوریکه به ازای هر دو میان‌بر p_ip_j و p_lp_m که p_ip_j میان‌بر p_lp_m را در بر می‌گیرد، داشته باشیم:

$$err(p_lp_m) \leq c.err(p_ip_j).$$

به عبارتی دیگر، یک تابع خطا برای یک مسیر c-یکنوا است اگر خطای هر میان‌بر از آن مسیر بیشتر از c برابر خطای میان‌برهایی که آن‌را دربرمی‌گیرند نباشد.

همچنین، تابع خطای err را برای یک مسیر e-تقریب‌پذیر می‌نامیم هرگاه به ازای هر میان‌بر p_ip_j که در الگوریتم بالا خطای آن توسط رویه تخمین خطا محاسبه می‌شود داشته باشیم:

$$err(p_ip_j) \leq err^*(p_ip_j) \leq e.err(p_ip_j).$$

به عبارت دیگر، بتوانیم خطای میان‌بر p_ip_j را با ضریب تقریب e محاسبه کنیم. در این‌صورت، این رویه تخمین خطا را e-تخمین‌گر می‌نامیم.

قضیه ۲۴ . فرض کنید تابع خطای بکاررفته در الگوریتم بالا بر روی مسیر اولیه c-یکنوا است و یک رویه e-تخمین‌گر برای آن وجود دارد. دراین صورت الگوریتم ساده‌سازی ارائه شده رای $l = 2k$ نسبت به $Opt(k)$، ce-رقابتی است. زمان مورد نیاز برای بهنگام‌سازی $\mathcal{Q}$ هنگام رسیدن یک راس جدید برابر با $O(\log k)$ بعلاوه زمان مورد نیاز برای بهنگام‌سازی داده‌ساختارهای رویه تخمین خطا است. علاوه بر حافظه مورد نیاز برای نگهداری $\mathcal{Q}$، حافظه مصرفی این الگوریتم برابر با $O(k)$ بعلاوه حافظه مورد نیاز رویه تخمین خطا است.

اثبات. مقدار دلخواه $n \geq 0$ را در نظر بگیرید و فرض کنید $\mathcal{Q}(n)$، $2k$-ساده‌سازی بدست آمده توسط این االگوریتم برای $\mathcal{P}(n)$ است. از آنجا که خطای $\mathcal{Q}(n)$ برابر با بیشینه خطای میان‌برهای آن است، کافی است نشان دهیم به ازای هر میان‌بر σ از $\mathcal{Q}(n)$ رابطه $err(\sigma) \leq c.e.err(\mathcal{Q}_{Opt(k)}(n))$ برقرار است.

فرض کنید میان‌بر σ هنگام پردازش نقطه p_m، برای $m \leq n$، در ساده‌سازی ظاهر شده است. اگر $m \leq 2k+1$ آنگاه $err(\sigma) = 0$ و رابطه مورد نظر برقرار است. در غیر اینصورت، فرض کنید $\mathcal{Q}(m-1) = \langle q_0, \ldots, q_{2k+1} \rangle$ $2k$-ساده‌سازی $\mathcal{P}(m-1)$ است. با رسیدن $p_m = q_{2k+2}$ راس $q_{2k+1} = p_{m-1}$ به $\mathcal{T}$ اضافه می‌شود. یک استدلال ساده شمارشی نشان می‌دهد که حداقل یکی از میان‌برهای $q_{t-1}q_{t+1}$ از $\mathcal{Q}(m-1)$، به ازای $1 \leq t \leq 2k+1$، که آن‌را با σ' نشان می‌دهیم، باید توسط یکی از حداکثر $k+1$ میان‌بر $\mathcal{Q}_{Opt(k)}(n)$ در بر گرفته شود. با توجه به این‌که در این مقطع میان‌بر σ دارای کمترین اولویت در میان میان‌برهای $\mathcal{T}$ است، خطای تخمین زده شده برای آن از خطای تخمین زده شده برای σ' کوچک‌تر است. بنابراین،

$$\begin{aligned} err(\mathcal{Q}_{Opt(k)}(n)) &\geq \tfrac{1}{c}err(\sigma') \geq \tfrac{1}{c\cdot e}err^*(\sigma') \\ &\geq \tfrac{1}{c\cdot e}err^*(\sigma) \geq \tfrac{1}{c\cdot e}err(\sigma). \end{aligned}$$

از رابطه بالا نتیجه می‌گیریم که الگوریتم ارائه شده نسبت به $Opt(k)$ که الگوریتم بهینه ساده‌سازی با k راس است، ce-رقابتی است. علاوه بر زمان و حافظه مورد نیاز رویه تخمین خطا،

در این الگوریتم $O(k)$ حافظه برای نگهداری صف اولویت و $O(\log k)$ زمان برای بهنگام‌سازی آن در پردازش یک راس جدید نیاز است. □

۹-۴ معیار خطای فاصله هاوس‌دورف

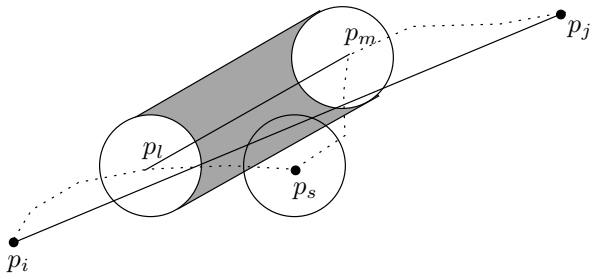
چنانچه گفته شد، الگوریتم ارائه شده فقط برای حالتی که تابع خطا بر روی مسیر مورد نظر یکنوا و تقریب‌پذیر باشد مناسب است. در این بخش نشان می‌دهیم که این ویژگی‌ها در مورد معیار خطای فاصله هاوس‌دورف بر روی مسیرهای محدب و xy-یکنوا برقرار است. یک مسیر محدب است اگر با اتصال فقط شروع و پایان آن یک چندضلعی محدب بوجود آید. یک مسیر xy-یکنوا است هرگاه هر خط افقی یا عمودی فقط در یک نقطه آن را قطع کند. لازم به ذکر است که در این دو نوع مسیر فاصله هاوس‌دورف و فاصله فِرشه بین هر میان‌بر p_ip_j و مسیر $\mathcal{P}(i,j)$ برابر است. بنابراین، نتایج بدست آمده برای این دو نوع مسیر که در ادامه این بخش بیان می‌شوند برای معیار خطای فاصله فِرشه نیز صدق می‌کنند. این نتایج نسبت به نتایج مربوط به معیار خطای فاصله فِرشه که در بخش ۹—۵ برای مسیرهای عمومی بیان می‌شوند قوی‌ترند.

لم زیر نتایج مربوط به یکنوا بودن گونه‌های مختلف مسیرها را بر اساس تابع خطای فاصله هاوس‌دورف بیان می‌کند.

لم ۳۲ . تابع خطای فاصله هاوس‌دورف بر روی مسیرهای محدب ۱-یکنوا و بر روی مسیرهای xy-یکنوا، ۲-یکنوا است. علاوه‌بر آن، هیچ مقدار ثابت c وجود ندارد که به ازای آن تابع خطای فاصله هاوس‌دورف بر روی یک مسیر x-یکنوا و در نتیجه یک مسیر عمومی c-یکنوا باشد.

اثبات. به‌سادگی می‌توان نشان داد که تابع خطای فاصله هاوس‌دورف بر روی مسیرهای محدب ۱-یکنوا است. همچنین با یک مثال ساده می‌توان نشان داد که برای هر مقدار c داده‌شده مسیری x-یکنوا وجود دارد که برای تابع خطای فاصله هاوس‌دورف c-یکنوا نباشد. یک نمونه از این مسیرها، یک مسیر زیگزاگی با چهار راس است که رئوس اول و سوم آن بسیار نزدیک به هم و رئوس دوم و چهارم نیز نزدیک هم قرار دارند.

حال مسیر xy-یکنوای $\mathcal{P} = \langle p_1, p_2, \ldots, p_n\rangle$ را درنظر بگیرید. فرض کنید p_ip_j و p_lp_m دو میان‌بر از این مسیر هستند که میان‌بر p_ip_j میان‌بر p_lp_m را دربر می‌گیرد و p_s نقطه‌ای روی مسیر $\mathcal{P}(l,m)$ است که $d(p_s, p_lp_m)$ برابر با خطای میان‌بر p_lp_m یعنی $err_H(p_lp_m)$ است. دایره‌های C_l، C_m و C_s را به شعاع $err_H(p_ip_j)$ و به ترتیب به مرکز p_l، p_m و p_s را که در شکل ۹—۱ نشان داده شده‌اند در نظر بگیرید. از آنجا که فاصله میانبر p_ip_j از نقاط p_l، p_m و p_s حداکثر برابر با $err_H(p_lp_m)$ است، این میان‌بر باید این سه دایره را قطع کند. فرض کنید p'_l، p'_m و p'_s به ترتیب نگاشت نقاط p_l، p_m و p_s بر روی میان‌بر p_ip_j باشد. واضح است که p'_l، p'_m و p'_s به ترتیب درون دایره‌های C_l، C_m و C_s قرار می‌گیرند. با توجه به اینکه $\mathcal{P}(i,j)$ یک مسیر xy-یکنوا است، p'_s بین



شکل ۹-۱: تابع خطای فاصله هاوس‌دورف برای هر مسیر xy-یکنوا، ۲-یکنوا است.

p'_l و p'_m قرار می‌گیرد که از آن رابطه زیر نتیجه می‌شود:

$$d(p'_s, p_l p_m) \leq \max(d(p'_l, p_l), d(p'_m, p_m)) \leq err_H(p_i p_j).$$

بنابراین،

$$\begin{aligned} err_H(p_l p_m) &= err_H(p_s, p_l p_m) \\ &\leq d(p_s, p'_s) + d(p'_s, p_l p_m) \\ &\leq ۲ err_H(p_i p_j). \end{aligned}$$

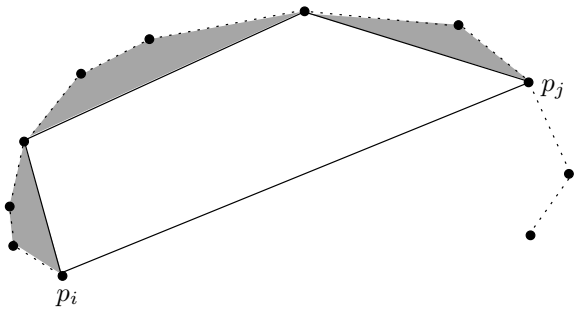
میان‌بر $p_i p_j$ ممکن است بر دایره‌های C_l، C_m و C_s مماس باشد که نشان می‌دهد ضریب یکنوایی ۲ کمترین مقدار ممکن و بسته[۶] است. □

قدم بعدی، تهیه رویه تخمین خطا برای مسیرهای محدب و xy-یکنوا است که در بخش‌های بعدی ارائه می‌شود. همچنین یک نتیجه منفی در مورد ساده‌سازی مسیرهای عمومی براساس معیار هاوس‌دورف و با حافظه ثابت ارائه می‌کنیم که نشان‌دهنده عدم وجود چنین الگوریتم‌هایی برای این حالت از مساله ساده‌سازی در حالت جویباری است.

۹-۴-۱ رویه تخمین خطا برای مسیرهای محدب

ایده رویه تخمین خطا برای مسیرهای محدب نگهداری تقریبی مساحت بین $p_i p_j$ و مسیر $\mathcal{P}(i,j)$ برای هر میان‌بر $p_i p_j$ است. فرض کنید $area(i,j)$ نشان دهنده این مساحت باشد. چنانچه دو زاویه $\angle p_{i+1} p_i p_j$ و $\angle p_{j-1} p_j p_i$ حداکثر ۹۰ درجه باشند، می‌توانیم تقریبی از $err_H(p_i p_j)$ را از روی $area(i,j)$ و $|p_i p_j|$ بدست آوریم. اگر $d_H(p_i p_j, \mathcal{P}(i,j))$ برابر با d باشد بیشینه مساحت بین $p_i p_j$ و $\mathcal{P}(i,j)$ مربوط به مستطیلی با قاعده $p_i p_j$ و ارتفاع d است و کمینه این مساحت مربوط به مثلثی با

[۶] Tight



شکل ۹-۲: مساحت‌هایی که توسط رویه تخمین خطا برای مسیرهای محدب نگهداری می‌شوند.

قاعده p_ip_j و ارتفاع d است. بنابراین،

$$d_H(p_ip_j, \mathcal{P}(i,j)) \leq ۲ \cdot area(i,j)/|p_ip_j| \leq ۲ \cdot d_H(p_ip_j, \mathcal{P}(i,j)),$$

و در نتیجه، $۲\,area(i,j)/|p_ip_j|$ می‌تواند به عنوان یک رویه ۲-تخمین‌گر برای $err_H(p_ip_j)$ استفاده شود. این روش برای حالتی که $\angle p_{i+۱}p_ip_j$ یا $\angle p_{j-۱}p_jp_i$ بزرگتر از ۹۰ درجه باشند قابل استفاده نیست. برای رفع این مشکل بصورت زیر عمل می‌کنیم.

برای هر میان‌بر p_ip_j که در ساده‌سازی جویباری استفاده شده است، مسیر $\mathcal{P}(i,j)$ را مطابق شکل ۹—۲ به حداکثر ۵ بخش تقسیم می‌کنیم. این تقسیم را در رئوسی از $\mathcal{P}(i,j)$ انجام می‌دهیم که بیشترین یا کمترین مقدار در راستای محور x یا y داشته باشند. چنانچه مثلا بیش از یک راس با کمترین مقدار x وجود داشته باشد، تقسیم بر روی اولین راس آنها انجام می‌شود. اطلاعاتی که برای هر میان‌بر p_ip_j نگهداری می‌شود شامل این نقاط تقسیم و مساحت بین هر یک از این بخش‌های $\mathcal{P}(l,m)$ و میان‌بر p_lp_m متناظر با آن است. چنانچه $\mathcal{P}(i,j)$ شامل چنین نقاط تقسیمی نباشد، فقط $area(i,j)$ نگهداری می‌شود.

مقدار $d_H(p_ip_j, \mathcal{P}(i,j))$ برابر با بیشینه مقادیر $d_H(p_ip_j, \mathcal{P}(l,m))$ روی همه قطعات $\mathcal{P}(l,m)$ است که $\mathcal{P}(i,j)$ به آنها شکسته شده است. بنابراین برای محاسبه $d_H(p_ip_j, \mathcal{P}(i,j))$ کافی است مقدار $d_H(p_ip_j, \mathcal{P}(l,m))$ برای این قطعات محاسبه شود. چون زاویه‌های $\angle p_{i+۱}p_ip_j$ و $\angle p_{j-۱}p_jp_i$ حداکثر ۹۰ درجه هستند، می‌توانیم برای تقریب $d_H(p_ip_j, \mathcal{P}(l,m))$ از مقدار

$$(۲ \cdot area(l,m)/|p_lp_m|) + d_H(p_ip_j, p_lp_m).$$

استفاده کنیم.

ادعا می کنیم که رویه بالا یک ۳-تخمین‌گر برای $err_H(p_ip_j)$ است. می‌دانیم:

$$\begin{aligned} d_H(p_ip_j, \mathcal{P}(l,m)) &\leq d_H(p_lp_m, \mathcal{P}(l,m)) + d_H(p_ip_j, p_lp_m) \\ &\leq \frac{۲ \cdot area(l,m)}{|p_lp_m|} + d_H(p_ip_j, p_lp_m). \end{aligned}$$

از طرف دیگر،

$$\begin{aligned}\mathbf{3}\cdot d_H(p_ip_j,\mathcal{P}(l,m)) &\geq \mathbf{3}\cdot\max(d_H(p_lp_m,\mathcal{P}(l,m)),\\ &\qquad d_H(p_ip_j,p_lp_m))\\ &\geq \tfrac{\mathbf{2}\cdot area(l,m)}{|p_lp_m|}+d_H(p_ip_j,p_lp_m).\end{aligned}$$

بنابر این $\mathbf{2}area(l,m)/|p_lp_m|)+d_H(p_ip_j,p_lp_m)$ یک تخمین با ضریب ۳ برای $d_H(p_ip_j,\mathcal{P}(l,m))$ است.

حال باید نشان دهیم که چگونه با رسیدن نقاط جدید و تغییر ساده‌سازی این اطلاعات را بهنگام‌سازی می‌کنیم. در مرحله دوم الگوریتم که می‌خواهیم $err^*_H(q_lq_{l+1})$ را محاسبه کنیم، اطلاعات مربوط به میان‌بر q_lq_{l+1} موجود است و رئوس q_{l+1} و q_{l+2} نیز دو نقطه متوالی در مسیر اولیه $\mathcal{P}$ هستند. بنابراین می‌توانیم اطلاعات مورد نظر را برای q_lq_{l+2} در $O(\mathbf{1})$ حساب کنیم. بصورت مشابه، می‌توانیم اطلاعات مورد نیاز در مرحله چهارم الگوریتم را محاسبه و بهنگام‌سازی کنیم که از بیان جزئیات ساده آن اجتناب می‌کنیم.

لم ۳۳ . یک رویه ۳-تخمین‌گر برای تابع خطای فاصله هاوس‌دورف برای مسیرهای محدب وجود دارد که حافظه مصرفی آن $O(k)$ و زمان بهنگام‌سازی آن برای هر نقطه جدید $O(\mathbf{1})$ است.

قضیه ۲۵ . یک الگوریتم ساده‌سازی جویباری برای مسیرهای محدب و معیار خطای فاصله هاوس‌دورف یا فِرشه وجود دارد که با نگهداری $\mathbf{2}k$ راس میانی، یک جویبار از نقاط ورودی را ساده‌سازی می‌کند و نسبت به $Opt(k)$ دارای ضریب رقابتی ۳ است. حافظه مورد نیاز این الگوریتم $O(k)$ است و هر نقطه جدید در زمان $O(\log k)$ پردازش می‌شود.

۹-۴-۲ رویه تخمین خطا برای مسیرهای xy-یکنوا

برای تخمین err_H در مسیرهای xy-یکنوا از مفهوم عرض[۷] نقاط استفاده می‌کنیم. عرض یک مجموعه نقاط در جهت $\overrightarrow{d}$ برابر با کمترین فاصله بین دو خط موازی در امتداد $\overrightarrow{d}$ است که همه نقاط بین آنها قرار می‌گیرند. فرض کنید $w(i,j)$ عرض نقاط $\mathcal{P}(i,j)$ در جهت $\overrightarrow{p_ip_j}$ است. از آنجا که $\mathcal{P}(i,j)$ یک مسیر xy-یکنوا است، پس بطور کامل درون مستطیلی قرار می‌گیرد که اضلاع آن موازی محورهای x و y و رئوس یک قطر آن نقاط p_i و p_j هستند. بنابراین،

$$\frac{w(i,j)}{\mathbf{2}}\leq err_H(p_ip_j)\leq w(i,j).$$

[۷] Width

در نتیجه، $w(i,j)$ را می‌توان به عنوان یک تخمین با ضریب تقریب ۲ برای $err_H(p_ip_j)$ استفاده کرد.

Agarwal و Yu [۸] یک الگوریتم جویباری برای نگهداری مجموعه هسته ارائه داده‌اند که برای تخمین عرض در هر جهت دلخواه کاربرد دارد. برای این‌کار، آنها یک ϵ–هسته با اندازه $O(\frac{1}{\sqrt{\epsilon}})$ از نقاط جویبار ورودی نگهداری می‌کنند و هر نقطه جدید در زمان سرشکنی[۸] $O(\log \frac{1}{\epsilon})$ پردازش می‌شود. سپس عرض نقاط در یک جهت دلخواه را می‌توان بصورت کارایی از روی پوش محدب مجموعه نقاط هسته، که با استفاده از داده‌ساختار پویای ارائه شده توسط Brodal و Jacob [۲۶] قابل نگهداری است، محاسبه کرد. داده‌ساختار اخیر دارای حافظه خطی و زمان بهنگام‌سازی لگاریتمی است. همچنین امکان گرفتن نقاط انتهایی در یک جهت دلخواه را دارد. بنابراین، می‌توانیم نقاط انتهایی[۹] را، که تعیین کننده عرض مجموعه نقاط است، در زمان $O(\log \frac{1}{\epsilon})$ محاسبه کنیم. تخمین عرض با استفاده از مجموعه هسته یک رویه با ضریب تقریب $\epsilon + 2$ فراهم می‌آورد.

لم ۳۴ . یک رویه $(\epsilon + 2)$–تخمین‌گر برای تابع خطای فاصله هاوس‌دورف بر روی مسیرهای xy-یکنوا وجود دارد که حافظه مصرفی آن $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ است و هر راس جدید را در زمان سرشکنی $O(k \log \frac{1}{\epsilon})$ پردازش می کند.

اثبات. الگوریتم ما فقط به مقدار تقریبی خطای میان‌برهای $q_{i-1}q_{i+1}$ نیاز دارد تا راس q_s را که باید از ساده‌سازی حذف شود، تعیین کند. ولی باید مجموعه هسته را برای همه میان‌برهایی که ممکن است در آینده در ساده‌سازی ظاهر شوند نگهداری کند تا بتواند خطای آنها را نیز تخمین بزند. این مجموعه میان‌برهاشامل همه میان‌برهای q_iq_j است که $0 \le i < j - 1 < 2k + 2$. بنابراین، لازم است که خطای تخمین زده شده برای $O(k^2)$ میان‌بر را نگهداری کنیم.

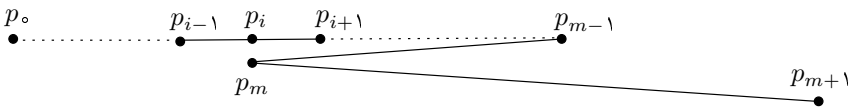
علاوه‌برآن، برای $O(k)$ میان‌بر q_ip_{n+1} باید مجموعه هسته مربوط به نقاط $\mathcal{P}(i, n+1)$ را نگهداری کنیم تا با دریافت راس جدید $q_{2k+2} = p_{n+1}$ بتوانیم خطای مربوط به میان‌برهای q_iq_{2k+2} برای $0 \le i \le 2k$ را تخمین بزنیم. مجموعه هسته میان‌بر q_ip_{n+1} را با افزودن راس p_{n+1} به مجموعه هسته مربوط به میان‌بر q_iq_{2k+1} مطابق الگوریتم Agarwal و Yu می‌سازیم.

در مجموع $O(\frac{k}{\sqrt{\epsilon}})$ حافظه برای نگهداری $O(k)$ مجموعه هسته و $O(k^2)$ حافظه برای نگهداری خطای $O(k^2)$ میان‌بر بالقوه مورد نیاز است. زمان بهنگام‌سازی هنگام رسیدن یک نقطه جدید شامل $O(k)$ درج در $O(k)$ مجموعه هسته موجود است. نقطه جدید در زمان سرشکنی $O(k \log \frac{1}{\epsilon})$ به این مجموعه هسته‌ها اضافه و خطای هر یک از این میان‌برهای بالقوه در زمان $O(\log \frac{1}{\epsilon})$ محاسبه می‌شود. بنابراین، زمان پردازش این تخمین‌گر خطا برای هر نقطه جدید برابر است با $O(k \log \frac{1}{\epsilon})$ و کل حافظه مورد نیاز آن $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ است. □

---

[۸] Amortized Time

[۹] Extreme Point



شکل ۹-۳: مولفه پایه برای نمایش عدم وجود رویه تخمین خطا برای معیار خطای فاصله هاوس‌دورف در حالت کلی.

قضیه ۲۶ . یک الگوریتم ساده‌سازی جویباری برای مسیرهای xy-یکنوا و معیار خطای فاصله هاوس‌دورف یا فِرشه وجود دارد که با نگهداری $2k$ راس میانی، یک جویبار از نقاط را ساده‌سازی می‌کند و نسبت به $Opt(k)$ دارای ضریب رقابتی $(4+\epsilon)$ است. حافظه مورد نیاز این الگوریتم برابر $O(k^2+\frac{k}{\sqrt{\epsilon}})$ است و هر نقطه جدید در زمان سرشکنی $O(k\log\frac{1}{\epsilon})$ پردازش می‌شود.

۹-۴-۳ معیار خطای فاصله هاوس‌دورف برای مسیرهای عمومی

در این بخش نشان می دهیم که برای تابع خطای فاصله هاوس‌دورف نمی‌توان یک الگوریتم ساده‌سازی جویباری با کمتر از $2k$ راس میانی ارائه داد بطوریکه ضریب رقابتی آن نسبت به $Opt(k)$ محدود باشد، مگر اینکه حافظه مصرفی آن $O(\frac{n}{k})$ باشد. در واقع این مطلب برای یک مسیر y-یکنوا نیز درست است.

قضیه ۲۷ . فرض کنید A یک الگوریتم جویباری است که یک $(2k-1)$-ساده‌سازی را برای مسیر مسطح عمومی $\mathcal{P}(n)$ محاسبه می کند و حداکثر می تواند $m-1$ نقطه از جویبار ورودی را ذخیره کند که $2k+2 \leq m \leq \frac{n}{k}$. برای هر $c>0$ و $km+1 \leq n$ یک مسیر y-یکنوای $p_0,p_1,\ldots,p_n$ وجود دارد که

$$err_H(\mathcal{Q}_{A(2k-1)}(n)) > c \cdot err_H(\mathcal{Q}_{Opt(k)}(n)).$$

اثبات. مولفه پایه‌ای این مسیر در شکل ۹—۳ نشان داده شده است که دارای ویژگی‌های زیر است:

الف) نقاط $p_0,p_1,\ldots,p_{m-1}$ هم راستا هستند.

ب) به ازای $0 \leq i \leq m-2$، رابطه $|p_ip_{i+1}| > c \cdot d_H(p_{m-1},p_ip_{m+1})$ برقرار است.

پ) برای $1 \leq i \leq m-1$، رابطه $d_H(p_{m-1},p_{i-1}p_{m+1}) > c \cdot d_H(p_{m-1},p_ip_{m+1})$ برقرار است.

برای بدست آوردن یک مسیر با ویژگی های بالا خط افقی l و نقطه p_{m-1} روی آن را در نظر می‌گیریم. نقطه p_{m+1} را در زیر l و در فاصله بسیار دور از p_{m-1} و در سمت راست آن قرار می‌دهیم بطوریکه فاصله p_{m+1} از l بزرگتر از $(c+\epsilon)^{m-1}$ باشد که $\epsilon \geq 0$ یک عدد به دلخواه

کوچک است. حال نقاط p_i به ازای $0 \leq i \leq m-2$ را روی l و در سمت چپ p_{m-1} بگونه‌ای قرار می‌دهیم که p_ip_{m+1} بر دایره‌ای به مرکز p_{m-1} و شعاع $(c+\epsilon)^{m-i-1}$ مماس باشد. با توجه به اینکه $d_H(p_{m+1}, l) > (c+\epsilon)$، نقطه p_i همواره وجود دارد.

حال الگوریتم ساده‌سازی A را در نظر بگیرید که حداکثر $m-1$ نقطه را می‌تواند ذخیره کند. با رسیدن نقطه p_{m-1}، A نمی‌تواند m نقطه را نگهداری کند و در نتیجه یکی از نقاط قابل دید باید از حافظه الگوریتم A پاک و بجای آن نقطه p_{m-1} ذخیره شود. فرض کنید p_i به ازای برخی i، $0 \leq i \leq m-2$، حذف می‌شود. حال، فرض کنید نقطه ورودی بعدی p_m زیر p_i و چسبیده به آن قرار دارد ($|p_ip_m| \approx 0$). تا اینجا، با انتخاب p_{m-1} در $\mathcal{Q}_{A(1)}(m)$ داریم: $err_H(\mathcal{Q}_{A(1)}(m)) = err_H(\mathcal{Q}_{Opt(1)}(m)) = 0$.

حال نقطه بعدی p_{m+1} را در نظر بگیرید که در محل متناظر با آن در مسیر ساخته شده بالا قرار دارد. از آنجا که در الگوریتم A نقطه p_i حذف شده است و دیگر قابل دسترسی نیست، پس $\mathcal{Q}_{A(1)}(m+1)$ بصورت p_0, p_j, p_{m+1} است که $i \neq j$.

سه حالت برای j وجود دارد:

۱) $0 \leq j < i$: بر اساس ویژگی (پ) داریم:

$$\begin{aligned} err_H(\mathcal{Q}_{A(1)}(m+1)) &= d_H(p_{m-1}, p_jp_{m+1}) \\ &> c \cdot d_H(p_{m-1}, p_ip_{m+1}) \\ &= c \cdot err_H(\mathcal{Q}_{Opt(1)}(m+1)). \end{aligned}$$

۲) $i < j \leq m-1$: بر اساس ویژگی (ب) داریم:

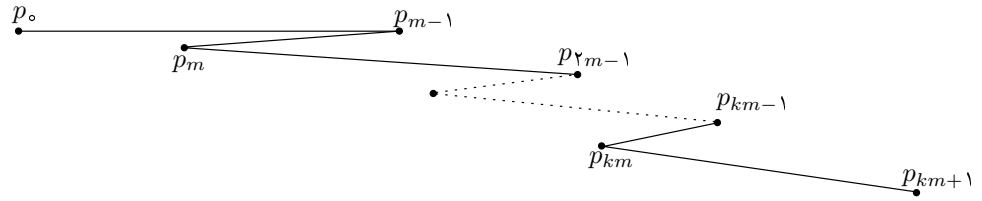
$$\begin{aligned} err_H(\mathcal{Q}_{A(1)}(m+1)) &\geq d_H(p_m, p_jp_{m+1}) \cong |p_ip_j| \\ &> c \cdot d_H(p_{m-1}, p_ip_{m+1}) \\ &= c \cdot err_H(\mathcal{Q}_{Opt(1)}(m+1)). \end{aligned}$$

۳) $j = m$: بر اساس ویژگی (الف) داریم:

$$\begin{aligned} err_H(\mathcal{Q}_{A(1)}(m+1)) &= d_H(p_{m-1}, p_0p_m) \cong |p_ip_{m-1}| \\ &> c \cdot d_H(p_{m-1}, p_ip_{m+1}) \\ &= c \cdot err_H(\mathcal{Q}_{Opt(1)}(m+1)). \end{aligned}$$

بنابراین، برای این‌که ضریب رقابتی الگوریتم A محدود باشد، باید حداقل دو نقطه p_{m-1} و p_m در ساده‌سازی A وجود داشته باشند که منجر به یک ۲-ساده‌سازی می‌شود.

می‌توانیم k مولفه از این نوع مسیر را بگونه‌ای پشت سرهم قرار دهیم که برای هر دو مولفه پشت سر هم، دو نقطه ابتدایی مولفه دوم بر روی دو نقطه انتهایی مولفه اول قرار بگیرند. این نوع ساختار در شکل ۹—۴ نشان داده شده است. به‌سادگی می‌توان نشان داد که علاوه بر نقاط اول و



شکل ۹-۴: یک مسیر که با ضریب رقابتی محدود ساده‌سازی نمی‌شود.

آخر این مسیر، الگوریتم A باید حداقل دو نقطه از هر یک از این مولفه‌ها را نگهداری کند تا ضریب رقابتی آن محدود باشد. در نتیجه،

$$err_H(\mathcal{Q}_{A(2k-1)}) > c \cdot err_H(\mathcal{Q}_{Opt(k)}(n)).$$

□

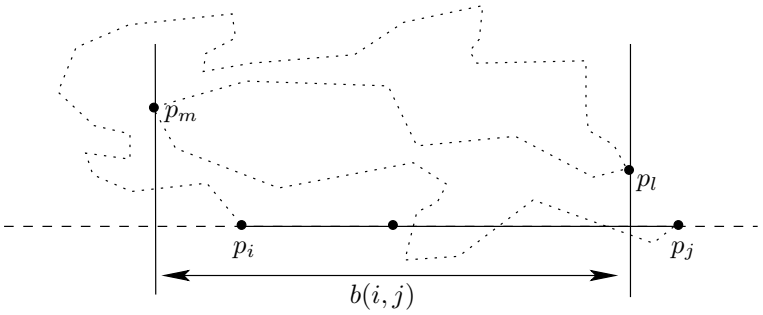
۹-۵ معیار خطای فاصله فِرشه

در این بخش نشان می‌دهیم که الگوریتم عمومی ارائه شده، برای معیار خطای فاصله فِرشه بر روی هر مسیر دلخواه، $O(1)$-رقابتی است. نخستین مساله اثبات یکنوا بودن این معیار خطا است که توسط Agarwal و سایرین [۵] اثبات شده است.

لم ۳۵ . میعار خطای فاصله فِرشه برای هر مسیر دلخواه، ۲-یکنوا است [۵].

پس از مشخص شدن یکنوا بودن این معیار خطا، باید رویه‌ای برای برآورد خطای میان‌برها ارائه کنیم. برای این کار از دو پارامتر برای تخمین زدن $err_F(p_ip_j)$ استفاده می‌کنیم. پارامتر اول، $w(i,j)$، عرض نقاط $\mathcal{P}(i,j)$ در راستای p_ip_j است که قبلا هم برای تخمین خطای فاصله هاوس‌دورف بر روی مسیرهای xy-یکنوا از آن استفاده کردیم. پارامتر دوم طول بزرگترین مسیر برگشتی در راستای p_ip_j است که به صورت زیر تعریف می‌شود. بدون کاهش کلیت بحث، فرض کنید p_ip_j بر روی یک خط افقی و p_j در سمت راست p_i قرار دارد. برای دو نقطه p_l و p_m بر روی مسیر $\mathcal{P}(i,j)$ که $l < m$، مسیر $\mathcal{P}(l,m)$ یک مسیر برگشتی بر روی $\mathcal{P}(i,j)$ است هر گاه $(p_m)_x < (p_l)_x$ که $(p)_x$ مولفه x نقطه p است. به عبارتی دیگر، $\mathcal{P}(l,m)$ نسبت به جهت $\overrightarrow{p_ip_j}$ یک مسیر برگشتی است اگر هنگام حرکت از p_l به p_m نسبت به جهت $\overrightarrow{p_ip_j}$ به عقب برگشته باشیم. به عنوان مثال، $\mathcal{P}(l,m)$ در شکل ۹—۵ یک مسیر برگشتی است.

طول مسیر برگشتی $\mathcal{P}(l,m)$ بر روی $\mathcal{P}(i,j)$ برابر با طول پاره‌خط مربوط به نگاشت p_lp_m بر روی خط گذران از p_i و p_j تعریف می‌شود که اگر p_ip_j افقی باشد مقدار آن برابر با $(p_l)_x - (p_m)_x$ است. بیشینه طول مسیرهای برگشتی روی $\mathcal{P}(i,j)$ را با $b(i,j)$ نشان می‌دهیم.



شکل ۹–۵: رابطه بین مسیرهای برگشتی و فاصله فِرشه.

لم ۳۶ . مقدار تابع خطای فاصله فِرشه را می توانیم با

$$\max(\frac{w(i,j)}{۲}, \frac{b(i,j)}{۲}) \leq err_F(p_ip_j) \leq ۲\sqrt{۲}\max(\frac{w(i,j)}{۲}, \frac{b(i,j)}{۲})$$

تخمین بزنیم.

اثبات. همانند بالا و بدون کاهش کلیت بحث، فرض می‌کنیم p_ip_j افقی است و p_j در سمت راست p_i قرار دارد. مشخص است که

$$err_F(p_ip_j) \geq err_H(p_ip_j) \geq \frac{w(i,j)}{۲}.$$

در نتیجه، برای اثبات قسمت اول کافی است نشان دهیم $err_F(p_ip_j) \leq \frac{b(i,j)}{۲}$. مسیر برگشتی $\mathcal{P}(i,j)$ بر روی $\mathcal{P}(l,m)$ مطابق شکل ۹–۵ را در نظر بگیرید که طول آن برابر با $b(i,j)$ است. فرض کنید نقطه r بر روی خط گذران از p_ip_j و در وسط نگاشت p_lp_m بر روی خط گذران از p_i و p_j قرار دارد. یعنی، r بر روی خط گذران از p_ip_j قرار دارد و مولفه x آن برابر با $((p_l)_x + (p_m)_x)/۲$ است. توجه داشته باشید که r لزوما بر روی پاره‌خط p_ip_j قرار نخواهد داشت. فاصله فِرشه بین p_ip_j و $\mathcal{P}(i,j)$ براساس یک زوج بهینه از تابع‌های پارامتری‌کننده روی p_ip_j و $\mathcal{P}(i,j)$ بدست می‌آید که برای هر نقطه p از $\mathcal{P}(i,j)$ یک نقطه $\overline{p}$ از p_ip_j را مشخص می‌کند بطوریکه اگر p قبل از q روی $\mathcal{P}(i,j)$ بیاید آنگاه $\overline{p}$ بعد از $\overline{q}$ روی p_ip_j نخواهد بود. حال مقادیر $\overline{pl}$ و $\overline{p_m}$ را در نظر بگیرید. اگر $\overline{pl}$ در سمت چپ r قرار داشته باشد آنگاه $|p_l\overline{pl}| \geq \frac{b(i,j)}{۲}$. از طرف دیگر، اگر $\overline{pl}$ در سمت راست یا بر روی r قرار داشته باشد آنگاه $\overline{p_m}$ نیز در سمت راست یا بر روی r قرار خواهد داشت که باز هم رابطه $|p_m\overline{p_m}| \geq \frac{b(i,j)}{۲}$ برقرار است. در نتیجه،

$$\max(\frac{w(i,j)}{۲}, \frac{b(i,j)}{۲}) \leq err_F(p_ip_j),$$

که قسمت اول لم را نتیجه می‌دهد.

برای اثبات قسمت دوم، باید نشان دهیم که $err_F(p_ip_j) \leq \sqrt{۲}\max(w(i,j),b(i,j))$. طبق تعریف رایج، معیار خطای فِرشه برحسب پیاده‌روی آدم و سگ بیان می‌شود که براساس آن باید یک زمانبندی برای این پیاده‌روی پیدا کنیم که آدم بر روی p_ip_j و سگ بر روی $\mathcal{P}(i,j)$ حرکت می‌کند و در این پیاده‌روی آدم و سگ هیچ‌گاه به عقب حرکت نمی‌کنند و فاصله آنها همواره کوچکتر یا مساوی $\sqrt{۲}\max(w(i,j),b(i,j))$ است.

یک پیاده‌روی با این ویژگی‌ها را می‌توانیم بصورت زیر طراحی کنیم. موقعیت آدم را با p_{man} و موقعیت سگ را با p_{dog} نشان می‌دهیم. ابتدا، $p_{man} = p_{dog} = p_i$. فرض کنید l خط عمودی گذران از p_i است. از میان همه نقاط برخورد l با $\mathcal{P}(i,j)$، p را دورترین نقطه در مسیر $\mathcal{P}(i,j)$ می‌گیریم. اگر p_i تنها نقطه برخورد باشد، p را برابر با p_i می‌گیریم. در پیاده‌روی پیشنهادی، سگ مسیر $\mathcal{P}(p_i,p)$ را طی می‌کند و در این مدت آدم روی نقطه p_i می‌ماند. فرض کنید q یک نقطه دلخواه روی مسیر $\mathcal{P}(p_i,p)$ است. در این صورت نقاط p_l و p_m، $l < m$، وجود دارند که $(p_l)_x \geq (p_i)_x$ و $(p_m)_x \leq (q)_x$. در نتیجه،

$$|(q)_x - (p_i)_x| \leq (p_l)_x - (p_m)_x \leq b(i,j).$$

همچنین، مشخص است که $|(q)_y - (p_i)_y| \leq w(i,j)$. بنابراین، در خلال این مرحله از حرکت رابطه زیر برقرار است:

$$|p_{man}p_{dog}| \leq \sqrt{۲}\max(w(i,j),b(i,j)).$$

پیاده‌روی را بدین ترتیب ادامه می‌دهیم: خط جاروب l را به سمت راست حرکت می‌دهیم. ابتدا، l مسیر $\mathcal{P}(p_i,p)$ را در یک نقطه قطع می‌کند. تا زمانی که این شرط برقرار است، p_{man} را برابر با $l \cap p_ip_j$ و p_{dog} را برابر با $l \cap \mathcal{P}(p,p_j)$ قرار می‌دهیم. در این مدت رابطه $|p_{man}p_{dog}| \leq w(i,j)$ برقرار است. با ادامه این پیاده‌روی، خط l ممکن است در برخی نقاط، $\mathcal{P}(p_{dog},p_j)$ را در نقاط دیگری غیز از p_{dog} قطع کند. دراین‌صورت، نقطه برخورد p را که در مسیر $\mathcal{P}(p_{dog},p_j)$ دورترین است در نظر می‌گیریم و اجازه می‌دهیم که سگ همه مسیر $\mathcal{P}(p_{dog},p)$ را طی کند، در حالی که آدم در موقعیت فعلی خود ثابت می‌ماند.

با استدلالی مشابه حالت آغازین حرکت، می‌توان نشان داد که در خلال این بخش از پیاده‌روی نیز رابطه $|p_{man}p_{dog}| \leq \sqrt{۲}\max(w(i,j),b(i,j))$ برقرار است.

در ادامه، خط l را به سمت راست حرکت می‌دهیم و آدم و سگ نیز مسیر خود را همانند بالا طی می‌کنند. این کار تا زمانی که l به p_j می‌رسد ادامه می‌یابد. در این لحظه آدم و سگ مسیرهای خود را پیموده و به مقصد p_j رسیده‌اند. بنابراین، در این پیاده‌روی پیشنهادی همواره رابطه $|p_{man}p_{dog}| \leq \sqrt{۲}\max(w(i,j),b(i,j))$ برقرار است که از آن اثبات قسمت دوم لم نتیجه می‌شود. □

مطابق لم ۳۶، برای تقریب زدن $err_F(p_ip_j)$ کافی است که مقدار $\max(w(i,j),b(i,j))$ تخمین زده شود. در بخش قبلی در ارائه تخمین خطای فاصله هاوس‌دورف برای مسیرهای xy—یکنوا نشان دادیم که چگونه می‌توان $w(i,j)$ را تخمین زد. در ادامه، روشی برای تخمین $b(i,j)$ ارائه می‌کنیم و نشان می‌دهیم که چگونه می‌توان این دو روش را با هم ترکیب کرد و رویه‌ای برای تخمین $err_F(p_ip_j)$ بدست آورد. لازم به ذکر است که اگر هیچ مسیر برگشتی وجود نداشته باشد، مقدار خطای فاصله فِرشه با مقدار خطای فاصله هاوس‌دورف برابر می‌شود و در نتیجه مسیرهای xy—یکنوا برای معیار خطای فاصله هاوس‌دورف حالت خاصی از مساله فعلی است.

در الگوریتم عمومی ارائه شده لازم است که (تخمینی از) مقدار خطای هر یک از میان‌برهای q_lq_{l+1} را در ساده‌سازی جاری داشته باشیم. به همین منظور، لازم است طول بزرگ‌ترین مسیر برگشتی در $\mathcal{P}(q_l,q_{l+1})$ را داشته باشیم و بتوانیم با رسیدن هر راس جدید آن‌را بهنگام‌سازی کنیم. عملیاتی که در پردازش یک راس جدید انجام می‌شوند شامل افزودن راس $q_{l+1}=p_{n+1}$ به انتهای ساده‌سازی و حذف راس q_s از ساده‌سازی است. برای این‌که بتوانیم b همه میان‌برها را بهنگام‌سازی کنیم اطلاعات زیر نگهداری می‌شوند.

برای سادگی بیان مطلب، فرض کنید که ما فقط می‌خواهیم طول بزرگترین مسیر برگشتی را در جهت مثبت محور x داشته باشیم. در این صورت برای هر میان‌بر p_ip_j از ساده‌سازی، اطلاعات زیر را نگهداری می‌کنیم :

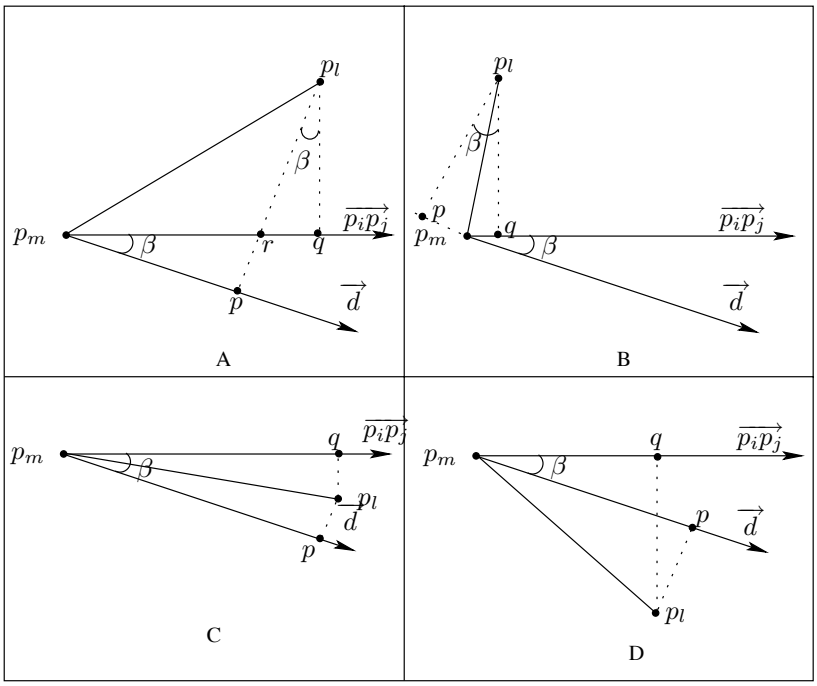
- $b(i,j)$، بیشینه طول مسیرهای برگشتی (نسبت به جهت مثبت محور x) روی مسیر $\mathcal{P}(i,j)$.
- $xmax(i,j)$، بیشینه مولفه x نقاط روی مسیر $\mathcal{P}(i,j)$.
- $xmin(i,j)$، کمینه مولفه x نقاط روی مسیر $\mathcal{P}(i,j)$.

حال میان‌بر q_lq_{l+1} را در نظر بگیرید و فرض کنید $q_l=p_i$، $q_{l+1}=p_t$ و $q_{l+1}=p_j$. دراینصورت، $b(i,j)$، یعنی بیشینه طول مسیرهای برگشتی روی مسیر $\mathcal{P}(q_l,q_{l+1})=\mathcal{P}(i,j)$، از رابطه زیر بدست می آید:

$$b(i,j)=\max(b(i,t),b(t,j),xmax(i,t)-xmin(t,j)).$$

در نتیجه، افزودن راس q_{l+1} به‌سادگی انجام می‌شود و کافی است سه مقدار بالا را برای هر میان‌بر $q_{l+1}q_{l+1}$ بدست آوریم. از آنجا که q_{l+1} و q_{l+1} دو نقطه متوالی از مسیر اصلی هستند، این سه مقدار برای آنها بصورت بدیهی موجود است. حذف راس q_s نیز به‌سادگی در زمان $O(1)$ انجام می‌شود (فرض کنید $q_{s-1}=p_i$ و $q_{s+1}=p_j$): طبق رابطه بالا، مقدار $b(i,j)$ را از روی مقادیر $b(i,s)$، $xmax(i,s)$ و $xmin(s,j)$ بدست می آوریم. محاسبه مقادیر $xmin(i,j)$ و $xmax(i,j)$ نیز بدیهی است.

بنابراین، در خلال اجرای الگوریتم می‌توانیم بیشینه طول مسیرهای برگشتی را نگهداری کنیم. اما، در استدلال بالا این فرض وجود داشت: در روش یاد شده $b(i,j)$ فقط در یک جهت ثابت (جهت



شکل ۹-۶: نمایش درستی رویه تخمین خطای معیار فِرشه.

مثبت محور x) نگهداری می‌شود. این در حالی است که ما باید بیشینه طول مسیر برگشتی را برای هر میان‌بر $q_iq_{i+۲}$ در جهت $\overrightarrow{q_iq_{i+۲}}$ نگهداری کنیم. این جهت‌ها برای میان‌برهای مختلف متفاوت است و از اول مشخص نیستند. برای حل این مشکل، تعداد $\frac{۲\pi}{\alpha}$ جهت کانونی و با فاصله یکسان از یکدیگر را به ازای مقدار مناسب $\alpha > ۰$ در نظر می‌گیریم و برای هر میان‌بر p_ip_j، اطلاعات گفته‌شده در بالا را در امتداد همه این جهت‌های کانونی نگهداری می‌کنیم.

حال، برای مقدار بیشینه طول مسیرهای برگشتی برای میان‌بر p_ip_j و در جهت $\overrightarrow{p_ip_j}$ از $b_{\overrightarrow{d}}(p_ip_j)$ استفاده می‌کنیم که بیشینه طول مسیرهای برگشتی $\mathcal{P}(i,j)$ در جهت $\overrightarrow{d}$ است. جهت $\overrightarrow{d}$ یکی از جهت‌های کانونی است که در جهت ساعت‌گرد نزدیکترین به $\overrightarrow{p_ip_j}$ است. در حالت کلی، استفاده از $b_{\overrightarrow{d}}(p_ip_j)$ ممکن است تقریب خوبی از $b(i,j)$ در جهت $\overrightarrow{p_ip_j}$، حتی برای مقادیر بسیار کوچک α، نباشد. اما این تقریب فقط برای حالاتی که $w(i,j)$ نسبت به $b(i,j)$ بزرگ باشد نامناسب است که در نتیجه با وجود نامناسب بودن تخمین $b(i,j)$، تقریب محاسبه شده برای $err_F(p_ip_j)$ هنوز مناسب باقی می‌ماند. این مساله در لم زیر بیان شده‌است.

لم ۳۷ . فرض کنید w و b به ترتیب عرض و بیشینه طول مسیر برگشتی برای نقاط مسیر $\mathcal{P}(i,j)$ در جهت $\overrightarrow{p_ip_j}$ هستند. همچنین، فرض کنید b^* بیشینه طول مسیرهای برگشتی روی $\mathcal{P}(i,j)$ در

جهت $\overrightarrow{d}$ است که $\overrightarrow{d}$ نزدیک‌ترین جهت در میان جهت‌های کانونی به $\overrightarrow{p_ip_j}$ در ترتیب ساعت‌گرد است. دراینصورت داریم:

$$b^* - \tan(\alpha) \cdot w \leq b \leq b^* + \tan(\alpha) \cdot (b^* + w).$$

اثبات. ابتدا نشان می‌دهیم که $b \leq b^* + \tan(\alpha) \cdot (b^* + w)$. فرض کنید مسیر برگشتی $\mathcal{P}(l, m)$ بیشترین طول را در جهت $\overrightarrow{p_ip_j}$ دارد. دو نیم‌خط خارج شده از نقطه p_m و به موازات $\overrightarrow{p_ip_j}$ و $\overrightarrow{d}$ را در نظر بگیرید. فرض کنید β زاویه بین این دو نیم‌خط است. با توجه به اینکه $\overrightarrow{d}$ نزدیک‌ترین جهت کانونی به $\overrightarrow{p_ip_j}$ در ترتیب ساعت‌گرد است، واضح است که $\beta \leq \alpha$. فرض کنید p و q به ترتیب تصاویر عمودی p_l بر روی خطوط گذران از p_m و در جهات $\overrightarrow{p_ip_j}$ و $\overrightarrow{d}$ هستند. براساس اینکه q_l بالا یا پایین خط گذران از p_m و در جهت p_ip_j قرار داشته باشد و نیز براساس اینکه p در سمت راست یا سمت چپ p_m در جهت $\overrightarrow{d}$ قرار گیرد، چهار حالت مختلف بوجود می‌آید که در شکل ۹—۶ نشان داده شده‌اند.

اثبات لم برای هر یک از این حالات به صورت زیر است:

(A)

$$\begin{aligned} b = |p_mq| &\leq |p_mp| + |pr| + |rq| \\ &\leq |p_mp| + |p_mp|\tan(\beta) + |p_lq|\tan(\beta) \\ &\leq b^* + \tan(\alpha) \cdot (b^* + w). \end{aligned}$$

(B)

$$\begin{aligned} b = |p_mq| &\leq |p_lq|\tan(\beta) \\ &\leq w\tan(\alpha) \\ &\leq b^* + \tan(\alpha) \cdot (b^* + w). \end{aligned}$$

(C)

$$\begin{aligned} b = |p_mq| &\leq |p_mp_l| \\ &\leq |p_mp| + |pp_l| \\ &\leq |p_mp| + |p_mp|\tan(\beta) \\ &\leq b^* + \tan(\alpha) \cdot (b^* + w). \end{aligned}$$

(D)

$$\begin{aligned} b = |p_mq| &\leq |p_mp| \\ &\leq b^* \\ &\leq b^* + \tan(\alpha) \cdot (b^* + w). \end{aligned}$$

رابطه $b^* - \tan(\alpha) \cdot w \leq b$ نیز با استدلالی مشابه اثبات می‌شود. □

رویه تخمین خطای نهایی err_F بصورت زیر تعریف می‌شود. فرض کنید w^* مقدار تقریبی عرض نقاط $\mathcal{P}(i,j)$ در جهت $\overrightarrow{p_ip_j}$ طبق الگوریتم ارائه شده توسط Agarwal و Yu است و b^* بیشینه طول مسیرهای برگشتی $\mathcal{P}(i,j)$ در جهت $\overrightarrow{d}$ است که $\overrightarrow{d}$ نزدیک‌ترین جهت کانونی به $\overrightarrow{p_ip_j}$ در ترتیب ساعت‌گرد است. عبارت زیر را به عنوان مقدار تقریبی خطای فاصله فِرشه در نظر می‌گیریم:

$$err^*_F(p_ip_j) = \sqrt{۲} \cdot \max(w^*, b^* + \tan(\alpha) \cdot (b^* + w^*)).$$

با ترکیب مطالب بالا و نتایج لم‌های ۳۶ و ۳۷، لم زیر را می‌توان ثابت کرد.

لم ۳۸ . مقدار $err^*_F(p_ip_j)$ یک $(۲\sqrt{۲}(۱+\epsilon)(۱+۴\tan(\alpha)))$-تقریب برای $err_F(p_ip_j)$ است. به عبارت دقیق‌تر،

$$err_F(p_ip_j) \leq err^*_F(p_ip_j) \leq ۲\sqrt{۲}(۱+\epsilon)(۱+۴\tan(\alpha)) \cdot err_F(p_ip_j).$$

اثبات. فرض کنید w و b به ترتیب برابر با $w(i,j)$ و $b(i,j)$ در جهت $\overrightarrow{p_ip_j}$ باشند. با توجه به اینکه w^* عرض مربوط به مجموعه هسته است، رابطه $w \leq w^* \leq (۱+\epsilon)w$ برقرار است. طبق لم ۳۶،

$$\begin{aligned} err_F(p_ip_j) &\leq ۲\sqrt{۲} \cdot \max(\tfrac{w}{۲}, \tfrac{b}{۲}) \\ &\leq \sqrt{۲} \cdot \max(w^*, b^* + \tan(\alpha) \cdot (b^* + w)) \\ &\leq \sqrt{۲} \cdot \max(w^*, b^* + \tan(\alpha) \cdot (b^* + w^*)) \\ &= err^*_F(p_ip_j). \end{aligned}$$

از طرف دیگر،

$$\begin{aligned} err^*_F(p_ip_j) &= \sqrt{۲} \cdot \max(w^*, b^* + \tan(\alpha) \cdot (b^* + w^*)) \\ &\leq \sqrt{۲} \cdot \max((۱+\epsilon)w, b + \tan(\alpha)w \\ &\quad + \tan(\alpha) \cdot (b + \tan(\alpha)w + (۱+\epsilon)w) \\ &\leq \sqrt{۲}(۱+\epsilon) \cdot \max(w, b + b\tan(\alpha) \\ &\quad + ۳w\tan(\alpha)) \\ &\leq \sqrt{۲}(۱+\epsilon)(۱+۴\tan(\alpha)) \cdot \max(w,b) \\ &\leq ۲\sqrt{۲}(۱+\epsilon)(۱+۴\tan(\alpha)) \cdot \max(\tfrac{w}{۲}, \tfrac{b}{۲}) \\ &\leq ۲\sqrt{۲}(۱+\epsilon)(۱+۴\tan(\alpha)) \cdot err_F(p_ip_j). \end{aligned}$$

□

چنانچه α و ϵ به اندازه کافی کوچک باشند، نتیجه نهایی مورد نظر، یعنی رویه‌ای با ضریب تقریب $O(۱)$ برای معیار خطای فاصله فِرشه، بدست می‌آید.

قضیه ۲۸ . یک الگوریتم جویباری برای محاسبه و نگهداری یک $2k$-ساده‌سازی از یک مسیر دلخواه و معیار خطای فاصله فِرشه وجود دارد که نسبت به الگوریتم بهینه $Opt(k)$ دارای ضریب رقابتی $4\sqrt{2}+\epsilon$ است. حافظه مورد نیاز این الگوریتم $O(k^2+\frac{k}{\sqrt{\epsilon}})$ است و هر نقطه ورودی در زمان سرشکنی $O(k\log\frac{1}{\epsilon})$ پردازش می‌شود.

۹-۶ معیار خطای فاصله‌ای وابسته به ناظر

در فصل ۸، دو معیار برای ساده‌سازی مرز ناحیه قابل دید از یک ناظر نقطه‌ای یا پاره‌خطی ارائه شد. همچنین الگوریتم‌هایی برای ساده‌سازی بر اساس این معیارها ارائه کردیم. در این بخش، ساده‌سازی چندضلعی قابل دید را در حالت جویباری بررسی می‌کنیم. در این حالت نقاط روی مرز ناحیه قابل دید بصورت جویباری دریافت می‌شوند و با دریافت هر نقطه می‌خواهیم مسیر ساده شده را برای این بخش از مرز ناحیه قابل دید ساده‌سازی کنیم.

برای این منظور، می‌خواهیم از الگوریتم عمومی ساده‌سازی ارائه شده در این فصل استفاده کنیم و ساده‌سازی را بر اساس معیار خطای فاصله‌ای وابسته به ناظر انجام دهیم. برای استفاده از این الگوریتم، باید نشان دهیم که معیار خطای فاصله‌ای وابسته به ناظر بر روی مرز ناحیه قابل دید یکنوا است و رویه‌ای برای تخمین خطای آن وجود دارد.

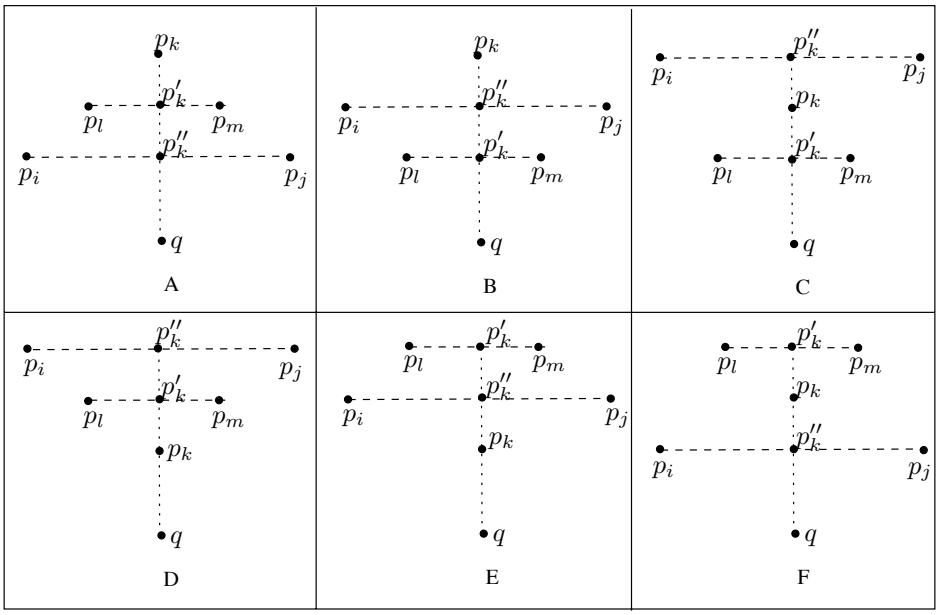
به‌سادگی می‌توانیم ثابت کنیم که معیار مساحتی وابسته به ناظر برای هیچ مقدار ثابتی یکنوا نیست. همچنین به‌سادگی قابل اثبات است که معیار خطای فاصله‌ای وابسته به ناظر برای چندضلعی قابل دید ضعیف یک ناظر پاره‌خطی (حتی برای ناظر واقع در یک چندضلعی ساده) نیز یکنوا نیست. ولی در ادامه این بخش نشان می‌دهیم که معیار خطای فاصله‌ای وابسته به ناظر برای مرز چندضلعی قابل دید ناظر نقطه‌ای و نیز چندضلعی قابل دید قوی ناظر پاره‌خطی یکنوا است و رویه‌هایی برای تخمین خطای آنها ارائه می‌کنیم.

۹-۶-۱ ساده‌سازی جویباری چندضلعی قابل دید ناظر نقطه‌ای

ابتدا نشان می‌دهیم که معیار خطای فاصله‌ای وابسته به ناظر برای چندضلعی قابل دید یک ناظر نقطه‌ای ۲-یکنوا است.

لم ۳۹ . معیار خطای فاصله‌ای وابسته به ناظر برای چندضلعی قابل دید ناظر نقطه‌ای q، یعنی $err_{dvis}(p_ip_j)$، ۲-یکنوا است.

اثبات. فرض کنید نقاط p_i، p_j، p_l و p_m روی $V(q)$ بگونه‌ای قرار دارند که $i\le l\le m\le j$ و $err_{dvis}(p_ip_j)$ برابر با خطای نقطه p_k روی $\mathcal{P}(l,m)$ است که $l\le k\le m$ و p'_k و p''_k به ترتیب محل



شکل ۹–۷: معیار خطای فاصله‌ای وابسته به ناظر بر روی چندضلعی قابل دید نقطه‌ای ۲–یکنوا است.

تقاطع خط گذرنده از qp_k و پاره‌خط‌های p_lp_m و p_ip_j است.

می‌دانیم، $V(q)$ یک چندضلعی ستاره‌ای است و q در هسته این چندضلعی قرار دارد. با توجه به ترتیب نقاط روی $V(q)$، خط‌های گذران از پاره‌خط‌های p_lq، p_mq و p_kq پاره‌خط p_ip_j را قطع می‌کنند و خط گذران از p_kq پاره‌خط p_lp_m را نیز قطع می‌کند. شش جایگشت مختلف برای موقعیت نقاط p_k، p'_k و p''_k روی خط گذران از qp_k وجود دارد که در شکل ۹—۷ نشان داده شده‌اند. برای همه این شش حالت روابط زیر را داریم:

$$\begin{array}{lcl} err_{vis}(p_ip_j) & \geq & \max(err_{vis}(p_l,p_ip_j), err_{vis}(p_m,p_ip_j), err_{vis}(p_k,p_ip_j)), \\ err_{vis}(p'_k,p_ip_j) & \leq & \max(err_{vis}(p_l,p_ip_j), err_{vis}(p_m,p_ip_j)). \end{array}$$

در نتیجه،

$$err_{vis}(p_ip_j) \geq \max(err_{vis}(p'_k,p_ip_j), err_{vis}(p_k,p_ip_j)).$$

بنابراین، برای اثبات لم باید صحت رابطه زیر را نشان دهیم:

$$\begin{array}{rcl} err_{vis}(p_lp_m) & = & err_{vis}(p_k,p_lp_m) \\ & \leq & ۲\max(err_{vis}(p'_k,p_ip_j), err_{vis}(p_k,p_ip_j)) \\ & \leq & ۲err_{vis}(p_ip_j). \end{array}$$

تساوی اول مربوط به این فرض است که p_k بین همه نقاط $\mathcal{P}(l,m)$ بیشترین خطا را روی p_lp_m دارد. نامساوی آخر نیز از دو رابطه بالاتر بدست می‌آید. بنابراین، کافی است که نامساوی وسطی را برای شش حالت نشان داده شده در شکل ۹—۷ نشان دهیم.

- حالت اول (قسمت A از شکل ۹—۷): در این حالت،

$$\begin{aligned} err_{vis}(p_k, p_lp_m) &= \frac{|p_kp'_k|}{|p_kq|} \leq \frac{|p_kp''_k|}{|p_kq|} = err_{vis}(p_k, p_ip_j) \\ &\leq \text{۲} \max(err_{vis}(p'_k, p_ip_j), err_{vis}(p_k, p_ip_j)). \end{aligned}$$

- حالت دوم (قسمت B از شکل ۹—۷): در این حالت اگر $|p_kp''_k| \geq |p''_kp'_k|$، آنگاه رابطه زیر را داریم:

$$\begin{aligned} err_{vis}(p_k, p_lp_m) &= \frac{|p_kp'_k|}{|p_kq|} = \frac{|p_kp''_k|+|p''kp'_k|}{|p_kq|} \leq \frac{\text{۲}|p_kp''_k|}{|p_kq|} \\ &\leq \text{۲} err_{vis}(p_k, p_ip_j) \leq \text{۲} \max(err_{vis}(p'_k, p_ip_j), err_{vis}(p_k, p_ip_j)), \end{aligned}$$

و اگر $|p_kp''_k| < |p''_kp'_k|$، آنگاه رابطه زیر را داریم:

$$\begin{aligned} err_{vis}(p_k, p_lp_m) &= \frac{|p_kp'_k|}{|p_kq|} \leq \frac{\text{۲}|p''_kp'_k|}{|p_kq|} \leq \frac{\text{۲}|p''_kp'_k|}{|p''_kq|} \\ &\leq \text{۲} err_{vis}(p'_k, p_ip_j) \leq \text{۲} \max(err_{vis}(p'_k, p_ip_j), err_{vis}(p_k, p_ip_j)). \end{aligned}$$

بنابراین، در هر دو حالت لم برقرار است.

- حالت سوم (قسمت C از شکل ۹—۷): در این حالت فرض کنید $|p_kp''_k| = x|p_kq| = x(|p_kp'_k| + |p'_kq|)$. دراین‌صورت،

$$\begin{aligned} \max(err_{vis}(p'_k, p_ip_j), err_{vis}(p_k, p_ip_j)) &\geq err_{vis}(p'_k, p_ip_j) \\ &= \frac{|p_kp'_k|+|p_kp''_k|}{|p_kp''_k|+|p_kq|} = \frac{|p_kp'_k|+x(|p_kp'_k|+|p'_kq|)}{x|p_kq|+|p_kq|} \\ &= \frac{(x+\text{۱})|p_kp'_k|+x|p'_kq|}{(x+\text{۱})|p_kq|} \\ &= err_{vis}(p_k, p_lp_m) + \frac{x|p'_kq|}{(x+\text{۱})|p_kq|} \\ &\geq err_{vis}(p_k, p_lp_m). \end{aligned}$$

- حالت چهارم (قسمت D از شکل ۹—۷): اثبات این حالت مشابه حالت سوم است.
- حالت پنجم (قسمت E از شکل ۹—۷): اثبات این حالت مشابه حالت دوم است.
- حالت ششم (قسمت F از شکل ۹—۷): اثبات این حالت مشابه حالت اول است.

بنابراین، در همه حالات نشان‌دادیم که $err_{vis}(p_lp_m) \leq \text{۲} err_{vis}(p_ip_j)$. همچنین، به‌سادگی می‌توان نشان داد که این مقدار (ضریب یکنوایی ۲) در حالات دوم و پنجم بسته است. □

حال یک رویه برای تخمین زدن خطای فاصله‌ای وابسته به ناظر q، $err_{dvis}(p_ip_j)$، برای استفاده در الگوریتم عمومی ارائه می‌کنیم. طبق لم ۲۴، برای محاسبه تقریبی $err_{dvis}(p_ip_j)$ کافی است که مقدار $d_o(q,p_ip_j)$، $w_L(i,j)$ و $w_U(i,j)$ را تخمین بزنیم. مقدار $d_o(q,p_ip_j)$ مشخص است. مقادیر w_L و w_U را همانند آنچه برای محاسبه خطای هاوس‌دورف روی مسیرهای xy-یکنوا و خطای فِرشه بیان کردیم، براساس روش ارائه شده توسط Agarwal و Yu [۸] تخمین می‌زنیم.

لم ۴۰ . یک رویه با ضریب تقریب $(1+\epsilon)$ وجود دارد که مقدار خطای فاصله‌ای وابسته به ناظر را برای هر میان‌بر از چندضلعی قابل دید یک ناظر نقطه‌ای تخمین می‌زند. حافظه مصرفی این رویه $O(k^2+\frac{k}{\sqrt{\epsilon}})$ است و هر نقطه ورودی در زمان سرشکنی $O(k\log\frac{1}{\epsilon})$ پردازش می شود.

اثبات. فرض کنید $\mathcal{Q}=\langle q_0,q_1,\ldots,q_{k+1}\rangle$ ساده‌سازی فعلی برای مسیر $\mathcal{P}(n)=\langle p_0,p_1,\ldots,p_n\rangle$ است. هر یک از این میان‌برهای q_iq_j برای $0\le i<j\le k+1$، ممکن است در ساده‌سازی‌های آینده ظاهر شوند و در نتیجه باید بتوانیم مقدار $err_{dvis}(p_ip_j)$ را تخمین بزنیم. برای این که بتوانیم این خطاها را تخمین بزنیم، برای هر میان‌بر q_iq_{k+1} یک مجموعه هسته نگهداری می‌کنیم که با استفاده از آن بتوانیم $w_U(i,k+1)$ و $w_L(i,k+1)$ را تخمین بزنیم. با داشتن مجموعه هسته مربوط به میان‌بر q_iq_{k+1}، دو نقطه انتهایی در جهت عمود بر q_iq_{k+1} بدست می‌آیند که فاصله این دو نقطه از q_iq_{k+1} برابر با مقدار تخمینی $w_U(i,k+1)$ و $w_L(i,k+1)$ با ضریب تقریب $1+\epsilon$ هستند.

درنتیجه، تعداد$O(k)$ مجموعه هسته نگهداری می‌کنیم که اندازه هر یک $O(\frac{1}{\sqrt{\epsilon}})$ است و برای تخمین خطای $err_{dvis}(p_ip_{k+1})$ استفاده می‌شوند. این مجموعه هسته‌ها به همان روشی که در لم ۲۴ آمده است نگهداری و بهنگام می‌شوند.

در مجموع، $O(\frac{k}{\sqrt{\epsilon}})$ حافظه برای نگهداری $O(k)$ مجموعه هسته و $O(k^2)$ حافظه برای نگهداری خطای مربوط به $O(k^2)$ میان‌بر بالقوه مورد نیاز است. بهنگام‌سازی داده‌ساختار این رویه به معنای افزودن نقطه جدید به مجموعه هسته‌های موجود است که در زمان سرشکنی $O(k\log\frac{1}{\epsilon})$ انجام می‌شود. در نتیجه، کل زمان مورد نیاز این رویه برابر است با $O(k\log\frac{1}{\epsilon})$. □

با ترکیب نتایج حاصل از لم‌های بالا و الگوریتم عمومی ساده‌سازی در حالت جویباری، نتیجه زیر در مورد ساده‌سازی چندضلعی قابل دید یک ناظر نقطه‌ای بر اساس معیار خطای فاصله‌ای وابسته به ناظر برقرار است.

قضیه ۲۹ . یک الگوریتم جویباری برای نگهداری یک $2k$-ساده‌سازی برای $V(q)$ بر اساس معیار خطای فاصله‌ای وابسته به ناظر وجود دارد که ضریب رقابتی آن نسبت به $Opt(k)$ برابر است با $(2+\epsilon)$. حافظه مورد نیاز این الگوریتم $O(k^2+\frac{k}{\sqrt{\epsilon}})$ است و هر نقطه ورودی در زمان سرشکنی $O(k\log\frac{1}{\epsilon})$ پردازش می‌شود.

۹-۶-۲ ساده‌سازی جویباری چندضلعی قابل دید قوی ناظر پاره‌خطی

مساله ساده‌سازی چندضلعی قابل دید قوی ناظر پاره‌خطی و معیار ساده‌سازی آن در فصل ۸ بررسی شد. معیار ساده‌سازی ارائه شده برای ساده‌سازی این مسیر یکنوا است و می‌توانیم برای تخمین خطای هر میان‌بر آن رویه‌ای کارا ارائه دهیم. درنتیجه، الگوریتم عمومی ساده‌سازی جویباری ارائه شده را می‌توانیم برای ساده‌سازی این نوع از مسیرها بکار ببریم.

لم ۴۱ . چندضلعی قابل دید قوی ناظر پاره‌خطی qr، $SV(qr)$، براساس معیار خطای فاصله‌ای وابسته به ناظر، ۲-یکنوا است.

اثبات. فرض کنید نقاط p_i، p_j، p_l و p_m چهار نقطه از $SV(qr) = \langle p_1, p_2, \ldots, p_n \rangle$ هستند که $i \leq l < m \leq j$. باید نشان دهیم که برای چنین وضعیتی رابطه $err_{svis}(p_l p_m) \leq 2.err_{svis}(p_i p_j)$ برقرار است.

طبق تعریف معیار خطای err_{svis}، روابط زیر برقرارند:

$$\begin{aligned} err_{svis}(p_l p_m) &= \max(err_{vis(q)}(p_l p_m), err_{vis(r)}(p_l p_m)), \\ err_{svis}(p_i p_j) &= \max(err_{vis(q)}(p_i p_j), err_{vis(r)}(p_i p_j)). \end{aligned}$$

همچنین می‌توانیم تصور کنیم که $SV(qr)$، چندضلعی قابل دید از ناظرهای نقطه‌ای q و r (برای دامنه‌های چندضلعی‌گونه مجازی) است. دراین صورت، طبق لم ۳۹، روابطزیر برقرارند:

$$\begin{aligned} err_{vis(q)}(p_l p_m) &\leq 2 err_{vis(q)}(p_i p_j), \\ err_{vis(r)}(p_l p_m) &\leq 2 err_{vis(r)}(p_i p_j). \end{aligned}$$

با ترکیب این رابطه‌ها، نتیجه زیر بدست می‌آید که موید درستی لم است:

$$err_{svis}(p_l p_m) \leq 2 \max(err_{vis(q)}(p_i p_j), err_{vis(r)}(p_i p_j)) = 2 err_{svis}(p_i p_j).$$

□

لم ۴۲ . یک رویه با ضریب تقریب $(1+\epsilon)$ برای ساده‌سازی چندضلعی قابل دید ناظر پاره‌خطی براساس معیار خطای فاصله‌ای وابسته به ناظر وجود دارد که حافظه مصرفی آن $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ است و هر نقطه ورودی در زمان سرشکنی $O(k \log \frac{1}{\epsilon})$ پردازش می‌شود.

اثبات. طبق لم ۴۰، یک رویه با ضریب تقریب $(1+\epsilon)$ برای تخمین $err_{vis(q)}(p_i p_j)$ و $err_{vis(r)}(p_i p_j)$ وجود دارد. فرض کنید $err^*_{vis(q)}(p_i p_j)$ و $err^*_{vis(r)}(p_i p_j)$ به ترتیب این مقدارهای

تقریبی برای $err_{vis(q)}(p_ip_j)$ و $err_{vis(r)}(p_ip_j)$ هستند. دراین‌صورت، روابط زیر وجود دارند:

$$\begin{aligned} err_{vis(q)}(p_ip_j) &\leq err^*_{vis(q)}(p_ip_j) \leq (1+\epsilon)err_{vis(q)}(p_ip_j), \\ err_{vis(r)}(p_ip_j) &\leq err^*_{vis(r)}(p_ip_j) \leq (1+\epsilon)err_{vis(r)}(p_ip_j). \end{aligned}$$

مقدار $err^*_{svis}(p_ip_j) = \max(err^*_{vis(q)}(p_ip_j), err^*_{vis(r)}(p_ip_j))$ را به عنوان مقدار تقریبی $err_{svis}(p_ip_j)$ می‌گیریم که روابط زیر در مورد آن برقرارند:

$$\begin{aligned} err_{svis}(p_ip_j) &= \max(err_{vis(q)}(p_ip_j), err_{vis(r)}(p_ip_j)) \leq err^*_{svis}(p_ip_j), \\ err^*_{svis}(p_ip_j) &\leq (1+\epsilon)\max(err_{vis(q)}(p_ip_j), err_{vis(r)}(p_ip_j)) = (1+\epsilon)err_{svis}(p_ip_j). \end{aligned}$$

در نتیجه، $err^*_{svis}(p_ip_j)$ یک مقدار تقریبی با ضریب تقریب $(1+\epsilon)$ برای $err_{svis}(p_ip_j)$ است. بدیهی است که پیچیدگی حافظه‌ای و زمانی این رویه تخمین خطا برابر با پیچیدگی حافظه‌ای و زمانی رویه ارائه شده برای ناظر نقطه‌ای است که در لم ۴۰ بیان شده است. □

با ترکیب نتایج حاصل از دو لم بالا و الگوریتم عمومی ساده‌سازی در حالت جویباری، نتیجه زیر در مورد ساده‌سازی چندضلعی قابل دید قوی ناظر پاره‌خطی برقرار است.

قضیه ۳۰ . یک الگوریتم جویباری برای نگهداری یک $2k$-ساده‌سازی برای $SV(qr)$ براساس معیار خطای فاصله‌ای وابسته به ناظر وجود دارد که ضریب رقابتی آن نسبت به $Opt(k)$ برابر است با $(2+\epsilon)$. حافظه مورد نیاز این الگوریتم $O(k^2 + \frac{k}{\sqrt{\epsilon}})$ است و هر نقطه ورودی در زمان سرشکنی $O(k\log\frac{1}{\epsilon})$ پردازش می‌شود.

۹-۷ نتیجه‌گیری

در این فصل، نخست یک الگوریتم جویباری ساده‌سازی مسیر ارائه شد که دارای ضریب رقابتی $O(1)$ برای مسیرهای محدب و xy-یکنوا بر اساس معیار خطای فاصله هاوس‌دورف، مسیرهای دلخواه بر اساس معیار خطای فاصله فِرشه و چندضلعی قابل دید نقطه‌ای و چندضلعی قابل دید قوی ناظر پاره‌خطی براساس معیار خطای فاصله‌ای وابسته به ناظر است.

در الگوریتم ارائه شده، چنانچه k (تعداد نقاط ساده‌سازی) را ثابت بگیریم، حافظه مصرفی آن $O(1)$ و زمان پردازش آن برای کل جویبار ورودی خطی و برای هر نقطه ورودی $O(1)$ است.

در الگوریتم ارائه شده، از ایده هم‌افزایی منابع استفاده کرده‌ایم: در الگوریتم ارائه شده، ساده‌سازی با $2k$ راس میانی انجام می‌شود ولی مقایسه کارایی آن با الگوریتم بهینه ساده‌سازی با k راس میانی صورت می‌گیرد.

نخستین پرسشی که در اینجا مطرح می‌شود این است که آیا ما می‌توانیم ساده‌سازی را با مقدار کمتر هم‌افزایی منابع یا بدون هم‌افزایی منابع انجام دهیم. ما نشان دادیم که برای مسیرهای عمومی و بر اساس معیار خطای فاصله هاوس‌دورف و بدون هم‌افزایی منابع نمی‌توانیم الگوریتم جویباری با ضریب رقابتی $O(1)$ داشته باشیم، ولی حتی با هم‌افزایی منابع نیز نتوانستیم الگوریتمی با ضریب رقابتی $O(1)$ برای چنین مسیر و معیار خطایی ارائه کنیم. بنابراین، یک فاصله بین نتیجه مثبت و منفی بدست آمده وجود دارد که رفع آن نیازمند مطالعه و بررسی بیشتر است.

یک جنبه دیگر مربوط به نتایج بدست آمده در این فصل، $O(k^2)$ حافظه مورد نیاز برای برخی مسیرها و معیارهای خطا است که ممکن است بتوان با حافظه کمتری به این نتایج رسید. به عنوان مثال، اگر بتوان مجموعه‌های هسته را بگونه‌ای با هم ادغام کرد که نیاز به نگهداری مجزای آنها نباشد، آنگاه می‌توان حافظه مورد نیاز را به $O(k)$ کاهش داد.

همچنین، الگوریتم عمومی ارائه شده را می‌توان به ابعاد بالاتر نیز تعمیم داد که البته این تعمیم روی ضریب تقریب و زمان و حافظه مصرفی نیز تاثیر خواهد داشت.

فصل ۱۰

نتیجه‌گیری

در این رساله، مساله‌ی تعیین و نگهداری فضای قابل دید از یک ناظر معرفی شده است. پس از تعریف مساله و گونه‌های مختلف آن، الگوریتم‌های پایه‌ای مربوط به حل نظری این مساله در حالت دوبعدی معرفی شده‌اند. سپس برای افزایش کارایی این الگوریتم‌ها در زمان پرس‌وجو، روش‌هایی معرفی شده‌اند که از طریق پیش‌پردازش و با نگهداری اطلاعاتی از وضعیت محیط مرجع، زمان پرس‌وجوی مناسبی را فراهم می‌آورند.

نگهداری دید یک ناظر نقطه‌ای در دامنه‌های چندضلعی‌گونه نیز مطالعه و برای آن دو الگوریتم ارائه شده است. همچنین، برای نگهداری فضای قابل دید از یک ناظر پاره‌خطی متحرک واقع در یک چندضلعی ساده یک الگوریتم ارائه شده است. برای محاسبه دید در فضای سه‌بعدی نیز داده‌ساختاری که اطلاعات دیداری محیط را در خود دارد، ارائه شده است که به کمک آن می‌توان دید یک ناظر را محاسبه کرد.

مساله ساده‌سازی مرز ناحیه قابل دید برای نگهداری ساده‌تر، یکی دیگر از مباحثی بوده که در این رساله به آن پرداخته شده است و برای حل آن در حالت عادی و نیز جویباری الگوریتم‌هایی ارائه شده است.

۱۰-۱ نتایج بدست آمده

نتایج بدست آمده به دو حوزه محاسبه-نگهداری دید و ساده‌سازی دید تقسیم می‌شوند. نتایج مربوط به حوزه اول شامل تعیین دید یک ناظر نقطه‌ای در دامنه‌های چندضلعی‌گونه [۱۳۹، ۱۴۴، ۱۴۶]، نگهداری دید یک ناظر نقطه‌ای متحرک در دامنه‌های چندضلعی‌گونه [۱۳۸، ۱۴۰، ۱۴۷]، نگهداری دید یک ناظر پاره‌خطی متحرک در چندضلعی‌های ساده [۸۴] و دید در فضای سه‌بعدی [۱۰۱] هستند.

نتایج بدست آمده در حوزه ساده‌سازی شامل تعیین معیار خطای وابسته به دید [۱۴۱، ۱۴۳]، ساده‌سازی چندضلعی قابل دید در حالت جویباری [۱۴۲، ۱۴۵] و ساده‌سازی در مدل جویباری [۱، ۲، ۳] هستند.

۱۰-۲ پژوهش‌های آینده و مسائل باز

در این بخش، مسائلی که در بررسی‌ها و مطالعات با آنها مواجه شده‌ایم، ولی در حوزه کار اصلی ما قرار نداشته‌اند و در نتیجه مورد بررسی دقیق قرار نگرفته‌اند یا برای حل آنها نتایج قابل قبولی بدست نیامده است، بیان می‌شوند. این مسائل به عنوان کارهای تحقیقی که در ادامه قابل انجام هستند مطرح می‌شوند و می‌توانند به عنوان پروژه‌های کارشناسی، کارشناسی ارشد و حتی دکتری انجام شوند.

ملاقات چندضلعی

این مساله حالت خاصی از مساله پیمایش تعدادی چندضلعی در صفحه با یک ترتیب خاص با قید کوتاه‌ترین مسیر پیمایش شده است که در [۳۸] مطرح و حل شده است. در این گونه از مسائل با محدود کردن چندضلعی‌ها به پاره‌خط به دنبال روش‌های بهتری نسبت به روش‌های موجود هستیم. بهترین نتیجه‌ای که تاکنون در ارتباط با این مساله حاصل شده است یک روش خطی برای این مساله در حالتی است که امتداد خطوط مربوط به این پاره‌خط‌ها پاره‌خط دیگری را قطع نکند. با این حال این مساله در حالت کلی باز است.

دید در حضور انعکاس

مساله دیگری که به صورت مختصر مورد توجه قرار گرفت، بررسی قابلیت دید در حضور انعکاس بود. نتایج موجود در این زمینه شامل انعکاس خطی یا موجی و نیز انعکاس در k مرحله است. با توجه به این‌که نتایج فعلی برای این مساله اندک و برای حالات متعددی غیربهینه هستند، احتمال اینکه بتوان مسائل خاصی را در این زمینه حل کرد، زیاد است.

دید دوطرفه

یک مساله مرتبط با محاسبه دید، تشخیص وضعیت دید دو شی نسبت به هم است. این مساله برای دو پاره‌خط بررسی و حل شده است [۱۰۲]. ولی، حل این مساله درحالت کلی و نیز در حالت سه بعدی هنوز باز است.

گراف قابلیت دید تقریبی

اندازه گراف قابلیت دید $O(n^2)$ است که استفاده از آن را در کاربردهای واقعی محدود می‌کند. ارائه روشی برای ساخت و نگهداری تقریبی گراف قابلیت دید و حل مساله نگهداری چندضلعی قابل دید در حالت‌های متحرک نتیجه‌ای ارزشمند و کاربردی خواهد بود.

نگهداری دید ناظر پاره‌خطی

این مساله برای ناظری که در یک چندضلعی ساده حرکت می‌کند، حل شده است [۸۴] ولی حل این مساله برای دامنه‌های چندضلعی‌گونه یک مساله باز است.

تعمیم ایده چندضلعی قابل دید دقیق

نتیجه بدست آمده در مورد چندضلعی قابل دید دقیق [۱۴۰] را می‌توان از دو جهت ادامه داد:

- پیاده‌سازی این روش و مقایسه آن با روشهای دیگر که می‌تواند میزان کارایی این روش را نسبت به روش‌های دیگر نشان دهد.
- تعمیم این روش به حالت ۲.۵ بعدی و TIN.

تعمیم الگوریتم‌های ساده‌سازی

مساله ساده‌سازی مسیر نیز از چند جهت از جمله موارد زیر قابل توسعه است:

- پیاده‌سازی الگوریتم ارائه شده [۱] و مقایسه آن با الگوریتم‌های مربوط به این مساله در حالت غیرجویباری.
- تلاش برای حل این مساله برای مسیرهای دلخواه با معیار خطای هاوس‌دورف.
- تلاش برای بهبود ضریب رقابتی الگوریتم یا اثبات بهینه بودن این الگوریتم.
- حل مساله ساده‌سازی برای معیارهای دیگر از جمله مساحت [۳۵].

افراز مبتنی بر قابلیت دید در فضای سه‌بعدی

یکی دیگر از مسائلی که در این پروژه مورد بررسی اولیه قرار گرفت افراز مبتنی بر قابلیت دید روی TIN است. این مساله برای چندضلعی‌های ساده و با مانع حل شده است [۲۴، ۱۳۹]. برای سطوح

سه بعدی از جمله در مورد TIN این مساله برای دو حالت بررسی شده است [۶، ۲۱، ۶۹]. حالت اول مربوط به افراز همه فضا و حالت دوم افراز دید در بینهایت است. همچنین روشی برای ساخت این نوع افراز ارائه شده است که بدون در نظر گرفتن تعداد ناحیه‌ها آنها را تعیین و نگهداری می‌کند [۵۲]. در این مساله، می‌خواهیم ناحیه‌های مربوط به افراز مبتنی بر قابلیت دید را روی سطح TIN تعیین کنیم که تاکنون روشی برای آن ارائه نشده است. در این مساله چند هدف می‌تواند مورد نظر باشد:

- تعیین تعداد ناحیه‌های با دید مختلف روی سطح TIN.
- ارائه الگوریتم و داده‌ساختار مناسب برای محاسبه این ناحیه‌ها و نگهداری دید مربوط به هر یک از آنها.
- بررسی دید یک نقطه دلخواه روی TIN.
- استفاده از افراز مبتنی بر قابلیت دید برای تعیین دید یک نقطه دلخواه روی TIN.
- تعیین و بهنگام‌سازی دید مربوط به یک ناظر متحرک روی TIN.
- بررسی این مساله روی TINهای خاص از جمله آنچه در [۹۷] تعریف شده است.

کوتاه‌ترین مسیر

هدف از این مساله تعیین کوتاهترین مسیر از یک نقطه مبدا تا جایی است که بتوان یک نقطه پرس‌وجوی دلخواه را در یک چندضلعی با مانع دید. این مساله در مورد چندضلعی ساده بصورت بهینه حل شده [۸۶]، ولی در مورد چندضلعی بامانع هنوز راه حلی برای آن ارائه نشده است. به نظر می‌رسد که داده‌ساختاری به نام ساعت شنی[1] که توسط Guibas و Hershberger [۶۱] ارائه شده است، قابل استفاده برای حل این مساله باشد که به هر حال نیازمند مطالعه و بررسی دقیق‌تر است.

محاسبه چندضلعی قابل دید در محیط‌های پویا یا سه‌بعدی

علاوه‌بر نقطه‌ی دید، ممکن است خود محیط و موانع موجود در آن نیز متحرک باشند. مطالعه این مساله و ارائه الگوریتمی کارا برای آن مسائلی باز هستند. علاوه‌برآن، بیشتر کارهای انجام شده برای تعیین دید در سه بعد مربوط به کاربردهای آن در گرافیک و پویانمایی است. در این کاربردها، دقت محاسبه و نیز سرعت انجام این کار متناسب با نوع کاربرد است. ارائه الگوریتم‌هایی بهینه و دقیق برای این مسائل یکی دیگر از مواردی است که از دیدگاه نظری ارزشمند است.

[1] Hourglass

ارائه الگوریتم‌های ترکیبی

افراز مبتنی بر قابلیت دید و مجتمع قابلیت دید دو نوع برخورد عمده برای پیش‌پردازش در مسائل قابلیت دید هستند که براساس آنها الگوریتم‌های متعددی ارائه شده است. در افراز مبتنی بر قابلیت دید، هزینه پیش‌پردازش بالا و در مقابل زمان پرس‌وجو پایین است، ولی در مجتمع قابلیت دید هزینه پیش‌پردازش پایین و زمان پرس‌وجو بالاست. مطالعه و تحقیق برای ارائه‌ی راه‌حل‌هایی که مبتنی بر ترکیب این دو روش باشند، می‌تواند منجر به نتایج ارزنده‌ای شود که از مزایای هر دو روش برخوردار باشند.

حل مسائل قابلیت دید با استفاده از داده‌ساختار جنبشی

یکی از حوزه‌های جدید کاری برای مسائل هندسه محاسباتی که در سال‌های اخیر فعال شده است، حل مسائل مربوط به محیط‌های پویا به کمک داده‌ساختارهایی است که تغییرات محیط را بصورت کارایی اعمال می‌کنند. این داده‌ساختارها را داده‌ساختار جنبشی می‌نامند.

استفاده از داده‌ساختارهای جنبشی باعث افزایش کارایی الگوریتم‌های مربوط به حل این مسائل در محیط‌های پویا می‌شود. در [۷۶] تلاش‌هایی برای ارائه الگوریتم مبتنی بر داده‌ساختار جنبشی برای حل مسائل قابلیت دید صورت گرفته است. با این حال، مسائل قابلیت دید زیادی وجود دارند که امکان حل کارای آنها با استفاده از داده‌ساختارهای جنبشی وجود دارد.

موازی‌سازی محاسبه قابلیت دید

حل مساله‌ی قابلیت دید با استفاده از الگوریتم‌های موازی نیز می‌تواند منجر به نتایج قابل توجهی شود. تاکنون فعالیت‌های عمده‌ای در این رابطه انجام نشده است و با توجه به قابلیت بالقوه موازی‌سازی الگوریتم‌های محاسبه دید، امکان دستیابی به نتایج قابل توجه در این زمینه دور از دسترس نیست.

پیوست A

مقالات استخراج شده از این رساله

نتایجی که از این رساله بدست آمده است در مقالات زیر به چاپ رسیده است یا برای چاپ ارسال شده است.

الف) مقالات چاپ شده یا ارسال شده برای مجلات بین‌المللی:

1) A. Zarei, M. Ghodsi, *Query Point Visibility Computation in Polygons with Holes*, Computational Geometry: Theory and Applications, 39 (2008), pp. 78–90.
2) M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei, *Streaming Algorithms for Line Simplification*, submitted to Discrete Computational Geometry, to appear.
3) A. Zarei, M. Ghodsi, *A practical approach for planar visibility mantenance*, International Journal for Geometry and Graphics, to appear.
4) A. Zarei, M. Ghodsi, *Observer-Dependent Simplification of Planar Visibility Polygon*, submitted to International Journal of Computational Geometry and Applications, submission date: 5/10/2008.

ب) مقالات چاپ شده یا ارسال شده برای کنفرانس‌های بین‌المللی:

5) A. Zarei and M. Ghodsi, *Efficient Computation of Query Point Visibility in Polygons with Holes*, 21st Annual ACM Symposium on Computational Geometry(SOCG), June 6-8, 2005, Pisa, Italy.

6) M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei, *Streaming Algorithms for Line Simplification under the Frechet Distance*, 23rd European Workshop on Computational Geometry, pp. 77–80, 2007.

7) M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei, *Streaming Algorithms for Line Simplification*, 23rd ACM Symp. on Computational Geometry (SoCG), pp.175–183, 2007.

8) A. A. Khosravi, A. Zarei, M. Ghodsi, *Efficient Visibility Maintenance of a Moving Segment Observer inside a Simple Polygon*, The 19th Canadian Conference on Computational Geometry (CCCG'2007), August 20–22, 2007.

9) A. Zarei, M. Ghodsi, *On Planar Visibility Polygon Simplification*, 24rd European Workshop on Computational Geometry, 2008.

10) A. Zarei, M. Ghodsi, *Simplifying Planar Visibility Polygons*, to appear, 8th Annual International Workshop on Computational Geometry and Applications (CGA'08), published by IEEE-CS, 2008.

11) A. Zarei, M. Ghodsi, *Efficient Observer-Dependent Simplification in Polygonal Domains*, submitted to 25th Annual ACM Symposium on Computational Geometry.

ب) مقالات چاپ شده در کنفرانس‌های داخلی:

12) A. Zarei and M. Ghodsi, *Some Results on Computing the Visibility of a Query Point inside Polygons with Holes*, 10th CSI Computer Conference (CSICC'2005), Iran Telecommunication Research Center, Tehran Feb 15–17, 2005, pp. 222-229.

13) A. Zarei, A. A. Khosravi, M. Ghodsi, *Maintaining Visibility Polygon of a Moving Point Observer in Polygons with Holes*, 11th CSI Computer Conference (CSICC'2006), IPM School of Computer Science, Tehran Jan 24–26, 2006.

14) M. NouriBygi, A. Zarei, M. Ghodsi, *Optimal Maintenace of 3D Visibility Complex*, 11th CSI Computer Conference (CSICC'2006), IPM School of Computer Science, Tehran, Jan 24–26, 2006.

15) A. Zarei, M. Ghodsi, *Exact Visibility Maintenance in Planar Polygonal Scenes in Practical Applications*, 12th CSI Computer Conference (CSICC'2007), Shahid Beheshti University, Tehran, Feb 20–22, 2007.

پیوست B

واژه‌نامه‌ی فارسی به انگلیسی

بازسازی شی ..... Object Reconstruction
برخط ..... Online
برنامه‌ریزی حرکت ..... Motion Planning
برنامه‌ریزی نقطه دید ..... Viewpoint Planning
بسته ..... Tight
پرتوافکنی ..... Ray-Shooting
پوش محدب ..... Convex Hull
پوشش خوشه‌ای ..... Clique Cover
تجزیه قابلیت دید ..... Visibility Decomposition
تشخیص شی ..... Object Recognition
تعقیب و گریز ..... Pursuit-Evasion
تعیین موانع ..... Occlusion Culling
تفکیک شعاعی ..... Radial Subdivision
تفکیک کنندگی ..... Separability
تقسیم و حل ..... Divide-and-Conquer
حذف سطوح مخفی ..... Hidden Surface Removal
حساس به خروجی ..... Output Sensitive
جویباری ..... Streaming
چندضلعی قابل دید ..... Visibility Polygon
چندضلعی قابل دید جزئی ..... Partial Visibility Polygon
چندضلعی قابل دید ضعیف ..... Weak Visibility Polygon
چندضلعی قابل دید قوی ..... Strong Visibility Polygon

خود مکان‌یابی ..... Self-Localization
داده‌ساختارهای جنبشی ..... Kinetic Data Structure
راس انعکاسی ..... Reflex Vertex
ردیابی اشعه ..... Ray-Tracing
زمان سرشکنی ..... Amortized Time
ساده‌سازی مسیر ..... Line Simplification
ساعت شنی ..... Hour-glass
سایه کامل ..... Hard Shadow
ضریب رقابتی ..... Compatitive Ratio
عرض ..... Width
غیر برخط ..... Offline
فضای تصویر ..... Image-Space
فضای شی ..... Object-Space
قطر برشی ..... Cut-Diagonal
قطعات آزاد بیشینه ..... Maximal Free Segments
گراف جنبه ..... Aspect Graph
گراف قابلیت دید ..... Visibility Graph
گراف مماس قابلیت دید ..... Tangent Visibility Graph
گوشه دید ..... Visible Cone
مجتمع قابلیت دید ..... Visibility Complex
مجموعه هسته ..... Core-Set
مدل ورودی جویباری ..... Streaming Input Model
موزه هنر ..... Art Gallery
مهرهای زمانی ..... Time-Stamp
نقشه توپولوژیکی ..... Topological Map
نورپردازی سراسری ..... Global Illumination
نیم‌سایه ..... Soft Shadow
هم‌افزایی منابع ..... Resource Augmentation
یکنواخت ..... Uniform

پیوست C

واژه‌نامه‌ی انگلیسی به فارسی

زمان سرشکنی ..... Amortized Time
موزه هنر ..... Art Gallery
گراف جنبه ..... Aspect Graph
پوشش خوشه‌ای ..... Clique Cover
ضریب رقابتی ..... Compatitive Ratio
پوش محدب ..... Convex Hull
مجموعه هسته ..... Core-Set
قطر برشی ..... Cut-Diagonal
تقسیم و حل ..... Divide-and-Conquer
نورپردازی سراسری ..... Global Illumination
سایه کامل ..... Hard Shadow
حذف سطوح مخفی ..... Hidden Surface Removal
ساعت شنی ..... Hour-glass
فضای تصویر ..... Image-Space
داده‌ساختارهای جنبشی ..... Kinetic Data Structure
ساده‌سازی مسیر ..... Line Simplification
قطعات آزاد بیشینه ..... Maximal Free Segments
برنامه‌ریزی حرکت ..... Motion Planning
تشخیص شی ..... Object Recognition
بازسازی شی ..... Object Reconstruction
فضای شی ..... Object-Space
تعیین موانع ..... Occlusion Culling

غیر برخط ..... Offline
برخط ..... Online
حساس به خروجی ..... Output Sensitive
چندضلعی قابل دید جزئی ..... Partial Visibility Polygon
تعقیب و گریز ..... Pursuit-Evasion
تفکیک شعاعی ..... Radial Subdivision
پرتو افکنی ..... Ray-Shooting
ردیابی اشعه ..... Ray-Tracing
راس انعکاسی ..... Reflex Vertex
هم‌افزایی منابع ..... Resource Augmentation
خود مکان‌یابی ..... Self-Localization
تفکیک کنندگی ..... Separability
نیم‌سایه ..... Soft Shadow
جویباری ..... Streaming
مدل ورودی جویباری ..... Streaming Input Model
چندضلعی قابل دید قوی ..... Strong Visibility Polygon
گراف مماس قابلیت دید ..... Tangent Visibility Graph
بسته ..... Tight
مهرهای زمانی ..... Time-Stamp
نقشه توپولوژیکی ..... Topological Map
یکنواخت ..... Uniform
برنامه‌ریزی نقطه دید ..... Viewpoint Planning
گوشه دید ..... Visible Cone
مجتمع قابلیت دید ..... Visibility Complex
تجزیه قابلیت دید ..... Visibility Decomposition
گراف قابلیت دید ..... Visibility Graph
چندضلعی قابل دید ..... Visibility Polygon
چندضلعی قابل دید ضعیف ..... Weak Visibility Polygon
عرض ..... Width

کتاب‌نامه

[1] M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei, *Streaming Algorithms for Line Simplification*, 23rd ACM Symp. on Computational Geometry (SoCG), pp.175–183, 2007.

[2] M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei, *Streaming Algorithms for Line Simplification*, Discrete Computational Geometry, to appear.

[3] M. A. Abam, M. de Berg, P. Hachenberger, and A. Zarei, *Streaming Algorithms for Line Simplification under the Frechet Distance*, 23rd European Workshop on Computational Geometry, pp. 77–80, 2007.

[4] A. L. Abbott, N. Ahuja, P.K. Allen, J. Aloimonos, M. Asada, R. Bajcsy, D.H. Ballard, B. Bederson, R. M. Bolle, P. Bonasso, C. M. Brown, P. J. Burt, D. Chapman, J. J. Clark, J. H. Connell, P. R. Cooper, J. D. Crisman, J. L. Crowley, M. J. Daily, J.O. Eklundh, F. P. Firby, M. Herman, P. Kahn, E. Krotkov, N. da Vitoria Lobo, H. Moraff, R. C. Nelson, H. K. Nishihara, T. J. Olson, D. Raviv, G. Sandini, and E. L. Schwartz, *Promising directions in active vision*, International Journal on Computer Vision, 1992.

[5] P.K. Agarwal, S. Har-Peled, N.H. Mustafa and Y. Wang, *Near-linear time approximation algorithms for curve simplification*, Algorithmica, 42:203–219, 2005.

[6] P. K. Agarwal and M. Sharir. *On the number of views of polyhedral terrains*, In Proc. 5th Canad. Conf. Comput. Geom., pages 55-60, 1993.

[7] P.K. Agarwal and K. R. Varadarajan, *Efficient algorithms for approximating polygonal chains*, Discrete and Compututationl Geometry 23(2):273–291, 2000.

[8] P.K. Agarwal and H. Yu, *A Space-Optimal Data-Stream Algorithm for Coresets in the Plane*, Proc. 23th ACM Symposium on Computational Geometry (SOCG), pages 1–10, 2007.

[9] H. Alt and M. Godau, *Computing the فرشه distance between two polygonal curves*, International Journal on Computational Geometry and Applications 5:75–91 1995.

[10] Arthur Appel, *Some techniques for shading machine renderings of solids*, In AFIPS 1968 Spring Joint Computer Conf., volume 32, pages 37-45, 1968.

[11] B. Aronov, L Guibas, M Teichmann and L. Zhang, *Visibility Queries and Maintenance in Simple Polygons*, Discrete and Computational Geometry 27(4), 2002, pp. 461-483.

[12] J. Arvo and D. Kirk, *A survey of ray tracing acceleration techniques*, In Andrew S. Glassner, editor, An introduction to ray tracing, pages 201–262. Academic Press, 1989.

[13] T. Asano, *Efficient Algorithms for Finding the Visibility Polygons for a Polygonal Region with Holes*, Manuscript, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1984.

[14] T. Asano, T. Asano, L. Guibas, J. Hershberger, H. Imai, *Visibility of disjoint polygons*, Algorithmica, 1 (1986), pp. 49-63.

[15] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, *Visibility Polygon Search and Euclidean Shortest Paths*, Proc. 26th FOCS, 1985, pp. 155-164.

[16] I. Ashdown, *Radiosity: A Programmers Perspective*, John Wiley & Sons, New York, NY, 1994.

[17] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice Hall, Englewood Cliffs, 2 edition, 1982.

[18] M. Baxandall, *Shadows and enlightenment*, Yale University Press, London, UK, 1995. (Ombres et Lumi‘eres, Gallimard).

[19] J. L. Bentley and Th. Ottmann, *Algorithms for Reporting and Counting Geometric Intersections*, IEEE Transactions on Computers, 28, pp. 643-647, 1979.

[20] M. de Berg, *Ray shooting, depth orders and hidden surface removal*, In Lecture Notes in Computer Science, volume 703. Springer Verlag, New York, 1993.

[21] M. de Berg, D. Halperin, M. Overmars and M. van Kreveld, *Sparse arrangements and the number of views of polyhedral scenes*, Internat. J. Comput. Geom. Appl. 7 (3) (1997) 175-195.

[22] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 1997.

[23] J. D. Boissonnat and M. Yvinec, *Algorithmic geometry*, Cambridge University Press, 1998.

[24] P. Bose, A. Lubiw and J. I. Munro, *Efficient Visibility Queries in Simple Polygons*, Computational Geometry: Theory and Applications, 23(3), pp. 313-335, 2002.

[25] M. Quinn Brewster, *Thermal Radiative Transfer & Properties*, John Wiley & Sons, New York, 1992.

[26] G. S. Brodal and R. Jacob, *Dynamic Planar Convex Hull*, Proc. 43rd Annual Symposium on Foundations of Computer Science (FOCS), pages 617–626, 2002

[27] L. Buzer, *Optimal Simplification of Polygonal Chain for Rendering*, 23rd ACM Symp. on Computational Geometry (SoCG), pages 168–174, 2007.

[28] W. S. Chan and F. Chin, *Approximation of polygonal curves with minimum number of line segments*, Proc. 3rd Annual International Symposium on Algorithms and Computing (ISAAC), LNCS 650, pages 378–387, 1992.

[29] B. Chazelle. *A theorem on polygon cutting with applications*, In Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci., pages 339-349, 1982.

[30] B. Chazelle, *The computational geometry impact task force report: An executive summary*, Lecture Notes in Computer Science, 1148:59-??, 1996.

[31] B. Chazelle and L. Guibas, *Visibility and Intersection Problems in Plane Geometry*, In Proc. 1th Annu. ACM Sympos. Comput. Geom., 1985, pp. 135-156.

[32] J. H. Clark, *Hierarchical geometric models for visible surface algorithms*, Communications of the ACM, 19(10):547-554, October 1976.

[33] M. F. Cohen and J. R. Wallace, *Radiosity and Realistic Image Synthesis*, Academic Press Professional, Boston, MA, 1993.

[34] B. I. L. Dalenback, *Room acoustic predictiom based on a unified treatment of diffuse and specular reflection*, Journal of the Acoustical Society of America, 100(2):899-909, August 1996.

[35] S. Daneshpajouh, A. Zarei, M. Ghodsi, *On Realistic Line Simplification under Area Measure*, submitted to ISAAC 2008.

[36] S. Dorward, *A survey of object-space hidden surface removal*, International Journal of Computational Geometry and Applications, 4(3):325-362, 1994.

[37] D. H. Douglas and T. K. Peucker, *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, Canadian Cartographer 10:112–122, 1973.

[38] M. Dror, A. Efrat, A. Lubiw, and JSB Mitchell, *Touring a sequence of polygons*, In Proc. 35th STOC, pp. 473–482, 2003.

[39] M. Drumheller, *Mobile robot localization using sonar*, IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-9(2):325-332, 1987. See also: S. B. Thesis, Dept. of Mechanical Engineering, MIT, 1984 and MIT AI Lab Memo BZ6, Mobile Robot Localization Osing Sonar.

[40] F. Durand, *3D Visibility: Analytical Study and Applications*, PhD thesis, University of Joseph Fourier, 1997.

[41] F. Durand, G. Drettakis, and C. Puech, *The 3D visibility complex*. ACM Transaction on Graphics, 21:2 (2002). pp. 176-206.

[42] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer Verlag, 1987.

[43] H. Edelsbrunner, and L. Guibas, *Topologically sweeping in an arrangement*, in Proceedings of the 18th Annual Symposium on Theory of Computing (1986), pp. 389-403.

[44] D. Eu and G.T. Toussaint, *On approximation polygonal cures in two and three dimensions*, CVGIP: Graphical Model and Image Processing, 56(3):231-246, 1994.

[45] O. Faugeras, *Three-dimensional computer vision*, MIT Press, Cambridge, MA, 1993.

[46] T. Feder and R. Motwani, *Clique partitions, graph compression, and speeding up algoithms*, In Proc. 27th Annual ACM Symposium on Theory of Computing, pages 123-133, 1991.

[47] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley Publishing Co., Reading, MA, 2nd edition, 1990. T 385.C587.

[48] Thomas Funkhouser, Ingrid Carlbom, Gary Elko, Gopal Pingali, Mohan Sondhi, and Jim West, *A beam tracing approach to acoustic modeling for interactive virtual environments*, In Michael F. Cohen, editor, Computer graphics: proceedings: SIGGRAPH 98 Conference proceedings, July 19-24, 1998, Computer Graphics -proceedings- 1998, pages 21-32, New York, NY 10036, USA and Reading, MA, USA, 1998. ACM Press and Addison-Wesley.

[49] S. Ghali and A. J. Stewart, *Incremental update of the visibility map as seen by a moving viewpoint in two dimensions*, In Seventh International Eurographics Workshop on Computer Animation and Simulation, pp. 1-11, August 1996.

[50] S. K. Ghosh and D. M. Mount, *An output sensitive algorithm for computing visibility graphs*, in Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science, Los Angeles, CA (1987), pp. 11-19. 59

[51] S. K. Ghosh and D. M. Mount, *An output-sensitive algorithm for computing visibility graphs*, SIAM Journal on Computing, Vol. 20, No. 5 (1991), pp. 888-910.

[52] Z. Gigus, J. Canny and R. Seidel, *Efficiently computing and representing aspect graphs of polyhedral objects*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, no. 6, pp. 542-551, June 1991.

[53] E.El Gindy, *An Efficient Algorithm for Computing the Weak Visibility Polygon from an Edge in Simple Polygons*, Technical Report, School of Computer Science, McGill University, Jan 1984.

[54] H.El Gindy and D. Avis, *A Linear Algorithm for Computing the Visibility Polygon from a Point*, Journal of Algorithms, 2, pp. 186-197, 1981.

[55] Andrew Glassner, *An introduction to raytracing*, Academic Press, Reading, MA, 1989.

[56] Andrew S. Glassner. Principles of Digital Image Synthesis. Morgan Kaufmann, San Francisco, CA, 1995.

[57] M. Godau, *A natural metric for curves: Computing the distance for polygonal chains and approximation algorithms*, Proc. 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS), pages 127–136, 1991.

[58] M. T. Goodrich, *Efficient piecewise-linear function approximation using the uniform metric*, Discrete and Computational Geometry 14:445–462, 1995.

[59] C. W. Grant, *Visibility Algorithms in Image Synthesis*, PhD thesis, U. of California, Davis, 1992.

[60] C. Guerra, *Vision and image processing algorithms*, In M. Atallah, editor, CRC Handbook of Algorithms and Theory of Computation. CRC Press, 1998.

[61] L. J. Guibas and J. Hershberger, *Optimal shortest path queries in a simple polygon*, J. Comput. Syst. Sci., 39:126-152, October 1989.

[62] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan, *Linear Time Algorithms for Visibility and Shortest Path Problems inside Simple Polygons*, Proc. Second Annual ACM Symp. on Computational Geometry, 1986, pp. 1-13.

[63] L.J. Guibas, J.E. Hershberger, J.S.B. Mitchell, and J.S. Snoeyink, *Approximating polygons and subdivisions with minimum link paths*, International Journal of Computatioanl Geometry and Applications 3:383–415, 1993.

[64] L. Guibas, R. Motwani, and P. Raghavan, *The Robot Localization Problem in two Dimensions*, SIAM J. of Computing 26, 4, 1997, pp. 1120-1138.

[65] E. Haines, *Ray tracing news*, http://www.povray.org/rtn/.

[66] S. L. Hakimi and E.F. Schmeichel, *Fitting polygonal functions to a set of points in the plane*, CVGIP: Graphical Models Image Processing 53:132–136, 1991.

[67] D. Halperin, L.E. Kavraki, and J.C. Latombe, *Robot algorithms*, In M. Atallah, editor, CRC Handbook of Algorithms and Theory of Computation. CRC Press, 1998.

[68] Dan Halperin, Lydia Kavraki, and Jean-Claude Latombe, *Robotics*, In J.E. Goodman and J. O'Rourke, editors, CRC Handbook of Discrete and Computational Geometry. CRC Press, 1997.

[69] D. Halperin and M. Sharir, *New bounds for lower envelopes in three dimensions, with applications to visibility in terrains*, Proc. 9th ACM Symp. on Computational Geometry, 1993, pp. 11–18.

[70] P. Heckbert and M. Garland, *Multiresolution modelling for fast rendering*, Proceedings of Graphics Interface'94), pages 43-50, 1994.

[71] P. J. Heffernan and J. S. B. Mitchell, *An Optimal Algorithm for Computing Visibility in the Plane*, SIAM Journal of Computing, 24(1):184–201, 1995.

[72] John Hershberger, *An Optimal Visibility Graph Algorithm for Triangulated Simple Polygons*, Algorithmica, 4(1): 141-155, 1989.

[73] J. Hershberger and J. Snoeyink, *An $O(n \log n)$ implementation of the Douglas-Peucker algorithm for line simplification*, Proc. 10th ACM Symposium on Computational Geometry (SOCG), pages 383–384, 1994.

[74] J. Hershberger and J. Snoeyink, *Cartographic line simplification and polygon CSG formulae in $O(n \log^* n)$ time*, In Proc. 5th International Workshop Algorithms and Data Structures (WADS), LNCS 1272, pages 93–103, 1997.

[75] J. Hershberger and S. Suri. *A pedestrain approach to ray shooting: Shoot a ray, take a walk*, J. Algorithms, 18:403-431, 1995.

[76] Olaf H. Holt, *Kinetic Visibility*, PhD THesis, Stanford University, USA, 2002.

[77] S. Hornus and C. Puech, *A Simple Kinetic Visibility Polygon*, In Proc. 18th EWCG'02, pp. 27-30 - 2002.

[78] Seth Hutchinson and Gregory D. Hager, *A tutorial on visual servo control*, IEEE Transactions on Robotics and Automation, 12(5):651-670, October 1996.

[79] H. Imai and M. Iri, *An optimal algorithm for approximating a piecewise linear function*, Journal of Information Processing 9(3):159–162, 1986.

[80] H. Imai and M. Iri, *Polygonal approximations of a curve-formulations and algorithms*, Computational Morphology, North-Holland, pages 71–86, 1988.

[81] , J. T. Kajiya, *The rendering equation*, In David C. Evans and Rusell J. Athay, editors, Computer Graphics (SIGGRAPH 86 Proceedings), volume 20(4), pages 143-150, August 1986.

[82] Kapoor, S. and S. N. Maheshwari, *Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles*, in Proceedings of the 4th Annual ACM Symposium on Computational Geometry, Urbana, IL, (1988), pp. 172-182.

[83] S. Kapoor and S. N. Maheshwari, *Efficiently constructing the visibility graph of a simple polygon with obstacles*, SIAM Journal on Computing, Vol. 30, No. 3 (2000), pp. 847-871.

[84] A. A. Khosravi, A. Zarei, M. Ghodsi, *Efficient Visibility Maintenance of a Moving Segment Observer inside a Simple Polygon*, to appear. The 19th Canadian Conference on Computational Geometry (CCCG'2007), August 20–22, 2007.

[85] D. Kirkpatrick, *Optimal Search in Planar Subdivisions*, SIAM Journal of Computing, 12, 1, pp. 28-35, 1983.

[86] C. Knauer, G. Rote, *Shortest Inspection-Path Queries in Simple Polygons*, Technical Report B-05-05, Institut fur Informatik, Freire Universitat Berlin, 2005.

[87] H. Kutruff, *Room Acoustics (3rd edition)*, Elseiver Applied Science, 1991.

[88] Jean-Claude Latombe, *Robot Motion Planning*, Kluwer, Dordrecht, The Netherlands, 1991.

[89] Lee, D. T., *Proximity and reachability in the plane*, Ph. D. thesis and Tech. Report ACT-12, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL (1978).

[90] D.T. Lee, *Visibility of a Simple Polygon*, Computer vision, Graphics, and Image Processing 22, 1983, pp. 207-221.

[91] D.T. Lee and F.P. Preparata, *Location of a Point in a Planar Subdivision and its Applications*, SIAM Journal of Computing, 6, 3, pp. 594-606, 1977.

[92] M. Lin and D. Manocha, *Applied computational geometry*, In Encyclopedia on Computer Science and Technology. Marcel Dekker, 1998.

[93] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes, *Real-time nonphotorealistic rendering*, In Turner Whitted, editor, SIGGRAPH 97 Conference Proceedings, Annual Conference Series, pages 415-420. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[94] Michael McKenna, *Worst-case optimal hidden-surface removal*, ACMTransactions on Graphics, 6(1):19-28, January 1987.

[95] A. Melkman, *On-line construction of the convex hull of a simple polyline*, Inform. Process. Lett., 25:11-12, 1987.

[96] A. Melkman and J. O'Rourke, *On polygonal chain approximation*, Computational Morphology, North-Holland, pages 87–95, 1988.

[97] E. Moet, Marc J. van Kreveld and A. F. van der Stappen, *On realistic terrains*, ACM Symposium on Computational Geometry pages 177-186, 2006.

[98] S.M. Muthukrishnan, *Data Streams: Algorithms and Applications*, http://athos.rutgers.edu/ muthu/stream-1-1.ps, 2003.

[99] Carl Mueller, *Architecture of image generators for flight simulators*, Technical Report TR-95-015, UNC Chapel Hill, February 1995.

[100] K. Nechvile and P. Tobola, *Local approach to dynamic visibility in the plane*, In Seventh Int. Conf. in Central Europe on Computer Graphics and Visualization, WSCG '99, February 1999.

[101] M. NouriBygi, A. Zarei, M. Ghodsi, *Optimal Maintenace of 3D Visibility Complex*, 11th CSI Computer Conference (CSICC'2006), IPM School of Computer Science, Tehran, Jan 24–26, 2006.

[102] M. NouriBygi, A. Zarei, M. Ghodsi, *Weak Visibility of Two Objects in Planar Polygonal Scenes*, 7th Annual International Workshop on Computational Geometry and Applications (CGA'07), Kuala Lumpur, Malaysia, August 26–29, 2007, LNCS by Springer-Verlag, 2007.

[103] Orti, R., Durand, F., Rivièr, S., and Puech, C. *Using the visibility complex for radiosity computation.* Proceeding ACM Workshop on Applied Computational Geometry, Philadelphia, May 1996

[104] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, NY, 1987.

[105] Joseph O'Rourke, *Computational geometry*, Cambridge University Press, 1994.

[106] Overmars, M. H., and E. Welzl, *New methods for constructing visibility graphs*, in Proceedings of the 4th Annual ACM Symposium on Computational Geometry, Urbana, IL (1988), pp. 164-171.

[107] Marco Pellegrini, *Ray-shooting and lines in space*, In J.E. Goodman and J. O'Rourke, editors, CRC Handbook of Discrete and Computational Geometry, pages 599-614. CRC Press, 1997.

[108] M. Pocchiola and G. Vegter, *Computing the visibility graph via pseudo-triangulations*, in Proceedings of the 11th Annual ACM Symposium on Computation Geometry, Vancouver, B.C., (1995), pp. 248-257.

[109] M. Pocchiola and G. Vegter, *The visibility complex*, Internat. J. Comput. Geom. Appl., 6(3):279–308, 1996.

[110] Arthur R. Pope, *Model-based object recognition: A survey of recent research*, Technical Report TR-94-04, University of British Columbia, Computer Science Department, January 1994.

[111] Michael Potmesil, *Generating octree models of 3D objects from their silhouettes in a sequence of images*, Computer Vision, Graphics, and Image Processing, 40(1):1-29, October 1987.

[112] F.P. Preparata, *A New Approach to Planar Point Location*, SIAM Journal of Computing, 10, 3, pp. 473-482, 1981.

[113] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[114] L. Richard Speer, *An updated cross-indexed guide to the ray-tracing literature*, Computer Graphics, 26(1):41-72, January 1992.

[115] Riviere, S., *Dynamic visibility in polygonal scenes with the visibility complex*, in Proceedings of the 13th Annual ACM Symposium on Computational Geometry, Nice, France, (1997), pp. 421-423.

[116] S. Rivi'ere, *Walking in the Visibility Complex with Applications to Visibility Polygons and Dynamic Visibility*, In Proc. Canadian Conf. on Comp. Geom., 1997.

[117] D.R. Roberts and A.D. Marshall, *A review of viewpoint planning*, Technical Report 97007, Univeristy of Wales, Cardiff, August 1997.

[118] David F. Rogers, *Procedural Elements for Computer Graphics*, Mc Graw-Hill, 2 edition, 1997.

[119] N. Sarnak and R. Tarjan, *Planar Point Location Using Persistent Search Trees*, Communications of the ACM, 29, 7, pp. 669-679, 1986.

[120] R. Seidel, *A Simple and Fast Incremental Randomized Algorithm for Computing Trapezoidal Decomposition and for Triangulating Polygons*, Comput. Geom. Theory Appl. 1:51-64, 1991.

[121] Thomas Shermer, *Recent results in art galleries*, In Proc. IEEE, pages 80:1384-1399, September 1992.

[122] Francois Sillion and Claude Puech, *Radiosity and Global Illumination*, Morgan Kaufmann, San Francisco, CA, 1994.

[123] Partha Srinivasan, Ping Liang, and Susan Hackwood, *Computational Geometric Methods in Volumetric Intersection for 3D Reconstruction*, In Proc. 1989 IEEE Int. Conf. Robotics and Automation, pages 190-195, 1989.

[124] S. Suri and J. O'Rourke, *Worst-Case Optimal Algorithms for Constructing Visibility Polygons with Holes*, In Proc. of the second annual symposium on Computational geometry, 1986, pp. 14-23.

[125] Ivan E. Sutherland, Robert F. Sproull, and Robert A. Schumacker, *A characterization of ten hidden-surface algorithms*, ACM Computing Surveys, 6(1):1-55, March 1974.

[126] Roberto Tamassia, Pankaj K. Agarwal, Nancy Amato, Danny Z. Chen, David Dobkin, Scot Drysdale, Steven Fortune, Michael T. Goodrich, John Hershberger, Joseph O'Rourke, Franco P. Preparata, Joerg- Rudiger Sack, Subhash Suri, Ioannis Tollis, Jeffrey S. Vitter, and Sue Whitesides, *Strategic directions in computational geometry*, ACM Computing Surveys, 28(4):591-606, December 1996.

[127] K.A. Tarabanis, P.K. Allen, and R.Y. Tsai, *A survey of sensor planning in computer vision*, IEEE Transactions on robotics and automation, 11(1):86-104, February 1995.

[128] G.T Toussaint, *On the complexity of approximating polygonal cures in the plane*, In Proc. IASTED International Symposium on Robotics and Automation, Lugano, Switzerland, 1985.

[129] Jorge Urrutia, *Art gallery and illumination problems*, In Jorg-Rudiger Sack and Jorge Urrutia, editors, Handbook on Computational Geometry. Elsevier Science, 1998.

[130] Eric Veach, *Robust Monte Carlo Methods for Light Transport Simulation*, Ph.d. thesis, Stanford University, December 1997.

[131] G. Vegter, *The visibility diagram: A data structure for visibility problems and motion planning*, In Proc. 2nd Scand. Workshop Algorithm Theory, volume 447 of Lecture Notes Comput. Sci., pages 97-110. Springer-Verlag, 1990.

[132] Alan Watt and Mark Watt, *Advanced Animation and Rendering Techniques: Theory and Practice*, Addison-Wesley, 1992.

[133] Welzl, E., *Constructing the visibility graph for n line segments in $O(n^2)$ time*, in Information Processing Letters, 20 (1985), pp. 167-171.

[134] Turner Whitted, *An improved illumination model for shaded display*, CACM, 1980, 23(6):343-349, 1980.

[135] G. Winkenbach and D. H. Salesin, *Computer-generated pen-and-ink illustration*, Computer Graphics, 28(Annual Conference Series):91-100, July 1994.

[136] Andrew Woo, Pierre Poulin, and Alain Fournier, *A survey of shadow algorithms* IEEE Computer Graphics and Applications, 10(6):13-32, November 1990.

[137] A. Zarei, M. Ghodsi, *A New Algorithm for Guarding Triangulated Irregular Networks*, to appear, CSI journal of Computer Science and Engineering, 2007.

[138] A. Zarei, M. Ghodsi, *A practical approach for planar visibility mantenance*, International Journal for Geometry and Graphics, to appear.

[139] A. Zarei and M. Ghodsi, *Efficient Computation of Query Point Visibility in Polygons with Holes*, 21st Annual ACM Symposium on Computational Geometry(SOCG), June 6-8, 2005, Pisa, Italy.

[140] A. Zarei, M. Ghodsi, *Exact Visibility Maintenance in Planar Polygonal Scenes in Practical Applications*, 12th CSI Computer Conference (CSICC'2007), Shahid Beheshti University, Tehran, Feb 20–22, 2007.

[141] A. Zarei, M. Ghodsi, *Efficient Observer-Dependent Simplification in Polygonal Domains*, submitted to 25th Annual ACM Symposium on Computational Geometry.

[142] A. Zarei, M. Ghodsi, *Observer-Dependent Simplification of Planar Visibility Polygon*, submitted to International Journal of Computational Geometry and Applications.

[143] A. Zarei, M. Ghodsi, *On Planar Visibility Polygon Simplification*, 24rd European Workshop on Computational Geometry, 2008.

[144] A. Zarei, M. Ghodsi, *Query Point Visibility Computation in Polygons with Holes*, Computational Geometry: Theory and Applications, 39 (2008), pp. 78–90.

[145] A. Zarei, M. Ghodsi, *Simplifying Planar Visibility Polygons*, to appear, 8th Annual International Workshop on Computational Geometry and Applications (CGA'08), published by IEEE-CS, 2008.

[146] A. Zarei and M. Ghodsi, *Some Results on Computing the Visibility of a Query Point inside Polygons with Holes*, 10th CSI Computer Conference (CSICC'2005), Iran Telecommunication Research Center, Tehran Feb 15–17, 2005, pp. 222-229.

[147] A. Zarei, A. A. Khosravi, M. Ghodsi, *Maintaining Visibility Polygon of a Moving Point Observer in Polygons with Holes*, 11th CSI Computer Conference (CSICC'2006), IPM School of Computer Science, Tehran Jan 24–26, 2006.

[148] H. Zarrabi-Zadeh and T. Chan, *A Simple Streaming Algorithm for Minimum Enclosing Balls*, In: Proc. 18th Canadian Conference on Computational Geometry (CCCG), pages 139–142, 2006.

Efficient Visibility Computation and Simplification in Different Environments

Abstract

In this thesis, we considered several types of the visibility problem. These problems include computing the visibility polygon of a point observer inside a polygonal domain, maintaining visibility polygon of a moving point observer, visibility coherence in space, maintaining visibility polygon of a moving segment observer and visibility dependent simplification. Furthermore, we considered these problems in both offline and streaming settings.

These problems arise in different practical areas, such as computer graphics, machine vision, robotics, motion planning, geographic information systems (GIS) and computational geometry.

We obtained effective theoretical results as well as superior practical algorithms. Our results are divided into two categories: visibility computation-maintenance and visibility simplification. Our results on the former category include efficient algorithms for computing the visibility polygon of a query point observer in 2D scenes, maintaining the visibility polygon of moving point or segment observers in 2D scenes, and a data structure for sketching visibility coherence between objects in space. For the latter, we proposed observer-dependent simplification criteria and proposed simplification algorithms for simplifying boundary of the visibility polygon of a point or segment observer in 2D for both streaming and offline input models. Moreover, we obtained a general streaming algorithm that can be used for classical line simplification problems.

Keywords: *Computational geometry, visibility problems, line simplification, streaming algorithms.*

Computer Engineering Department

Sharif University of Technology

Efficient Visibility Computation and Simplification in Different Environments

Submitted in Partial Fulfillment
of the Requirements for the Degree of

Ph.D.
in
Computer Engineering (Software)

by

Alireza Zarei

Supervisor
Professor Mohammad Ghodsi

November 2008