شبیه سازی حرکت وسیله نقلیه

فهرست مطالب

مقدمه

آزمایشات واقعی خودروهای جاده ای وقت گیر ، پر زحمت و پرهزینه است و چندین نگرانی ایمنی را شامل می شود. شبیه سازهای حرکت وسایل نقلیه جاده ای (RVMS) می توانند به آزمایش وسیله نقلیه کمک کرده و مشکلات مربوط به انجام آزمایشات خودرو را حذف یا کاهش دهند. با این حال ، چنین شبیه سازهایی باید سطح بالایی از دقت را به منظور ارائه نتایج واقعی و قابل اعتماد از خود نشان دهند. در این تحقیق ، ما RVMS موجود را مرور کرده و هر یک از زیر سیستم های اصلی RVMS مربوط به تحقیق و توسعه دینامیک خودرو را مورد بحث قرار می دهیم. تولیدکنندگان خودرو باید طیف وسیعی از معیارها را در طول طراحی و توسعه خودرو از جمله معیارهای مربوط به ایمنی ، سواری و هندلینگ رعایت کنند.پارامترهای مرتبط معمولاً از طریق رویکردهای تحلیلی ، شبیه سازی و تجربی مورد بررسی قرار می گیرند. آزمایشات ذهنی وسیله نقلیه جاده ای بخش مهمی از تأیید و تأیید سیستم ها هستند و معمولاً برای اصلاح و بهبود عملکرد خودرو استفاده می شوند. با این حال ، آزمایش های واقعی خودروهای جاده ای وقت گیر ، پرهزینه و ذاتاً نگرانی های ایمنی زیادی را ایجاد می کند.

شبیه ساز حرکت وسایل نقلیه جاده ای (RVMS) می تواند موانع مربوط به انجام آزمایشات واقعی خودرو را حذف یا کاهش دهد. شبیه سازهای حرکتی به مهندسان این امکان را می دهند تا سناریوهای متعددی را در یک سیستم به سرعت و به راحتی آزمایش کنند و در زمان و هزینه قابل توجهی صرفه جویی شود. علاوه بر این ، شبیه سازها به مهندسان این امکان را می دهند که بر روی یک راه حل طراحی خاص تمرکز کنند بدون اینکه حواس پرتی ناشی از عوامل محیطی یا لوازم جانبی غیر مرتبط خودرو را پرت کنند. در غیاب تأثیرات خارجی ، مهندسان قادرند یک پارامتر خاص را سریعتر و با دقت بیشتر از آنچه در آزمایش خودروهای واقعی به دست می آید تنظیم کنند. شبیه سازهای حرکتی باید راستی و دقت بالایی را ارائه دهند تا نتایج آزمایش خودرو واقعی و قابل اعتماد را بدست آورند. طیف وسیعی از اپراتورهای واقعیت مجازی وجود دارد که ممکن است به شبیه سازهای حرکت برای تکرار قابلیت ها خودرو کمک کند. اینها در درجه اول شامل نشانه های حرکتی ، بصری ، صوتی و لمسی هستند که با توجه به کاهش فاصله حرکت و آستانه درک انسان تنظیم می شوند.

شبیه سازهای حرکتی در ابتدا برای شبیه سازی وسیله نقلیه پرواز طراحی شده و مورد استفاده قرار گرفتند ، جایی که آزمایش های گران و خطرناک به راحتی کاربردهای آنها را توجیه می کند. بعدها ، سازندگان خودرو و موسسات تحقیقاتی شبیه سازهایی را برای تحقیق و توسعه وسایل نقلیه جاده ای توسعه دادند. با این حال ، RVMS لزوماً پیچیده تر از شبیه سازهای حرکت هوایی نیست. شاید به دلیل حرکات فرکانس بالا ، مانورهای سریع و فعل و انفعالات محیطی مرتبط با وسایل نقلیه جاده ای ، آنها از نظر ساختار و الگوریتم طراحی پیچیده تر باشند. کاربردهای اصلی RVMS در طول زمان مطالعات عوامل انسانی ، آموزش رانندگان ، توسعه زیرسیستم خودرو و بررسی ویژگی های جاده و ترافیک است.

شبیه ساز حرکت وسایل نقلیه جاده ای

در این مقاله ، ما RVMS را بر اساس راستی ، قابلیت استفاده ، پیچیدگی و هزینه آنها طبقه بندی می کنیم. این طبقه بندی به سه نوع RVMS منجر می شود که عبارتند از: RVMS سطح بالا ، سطح متوسط و سطح پایین. این روش برای طبقه بندی شبیه سازهای حرکت توسط ویر و کلارک معرفی شده است و بیشتر توسعه یافته است و اغلب توسط نویسندگان دیگر مورد استفاده قرار می گیرد. در اینجا معیارهای طبقه بندی در نظر گرفته شده و از نظر کیفی مورد استفاده قرار می گیرند زیرا دسترسی به جزئیات RVMS موجود محدود شده است.

سطح بالا - کلاس جهانی

RVMS سطح بالا یا "کلاس جهانی" دارای یک پلت فرم ساختار یافته ترکیبی است و بالاترین سطح قابلیت استفاده را برای یک برنامه خاص و بالاترین سطح راستی در بازتولید همه نشانه های بازخورد حرکتی ، بصری ، شنیداری و نیرو ارائه می دهد. آنها قادر به شبیه سازی کامل دینامیک وسایل نقلیه هستند و میدان دید وسیعی را ارائه می دهند. RVMS سطح بالا معمولاً شامل یک کابین کامل با تمام قابلیت ها است که بر روی پلت فرم حرکت با حداقل 6 DOF نصب شده است.

این شبیه سازها معمولاً توسط سازندگان خودرو برای طراحی ، آزمایش و تأیید سیستم های جدید مانند سیستم های کنترل ایمنی پایداری خودرو (VSC) و سیستم های کروز کنترل تطبیقی (ACC) استفاده می شوند. مشهورترین شبیه سازهای این دسته ، شبیه ساز پیشرفته ملی رانندگی (NADS) در IOWA ، ایالات متحده هستند.

سطح متوسط شبیه ساز های حرکت عمومی

شبیه سازهای سطح متوسط عموماً از کاربرد بالایی با تمرکز بر کاربرد جهانی و سطح بالای وفاداری در بازتولید برخی از نشانه های بازخورد اینرسی ، بصری ، شنیداری و نیروی برخوردار هستند. ساختار این شبیه سازها شامل یک کابین جهانی و جزئی با سایر زیر سیستم های مربوطه است که در یک پلت فرم حرکت گنجانده شده است. این شبیه سازها در مقایسه با RVMS سطح بالا دارای ساختار ساده تری هستند. با این حال ، با پیکربندی مناسب و هماهنگ سازی نشانه های حرکت و واقعیت مجازی ، RVMS سطح متوسط می تواند طیف وسیعی از سناریوها و وظایف رانندگی را تکرار کرده و یک غوطه وری واقعی ملموس ایجاد کند.

علاوه بر این ، هزینه کمتر طراحی ، توسعه و شبیه سازی RVMS سطح متوسط برای بسیاری از برنامه های تحقیقاتی مورد توجه است. به همین دلیل است که از شبیه سازهای سطح متوسط به جای شبیه سازهای سطح بالا در چندین مکان استفاده می شود. از منظر دینامیک خودرو ، این شبیه سازها عموماً قادر به بازتولید هستند: (1) ارتعاش با فرکانس بالا و (2) رفتار در پیچ خودرو که به ترتیب برای راحتی سواری و عملکرد رانندگی خودرو بسیار مهم است.

در مقاله حاضر ، تمرکز اصلی بر شبیه سازهای سطح متوسط است. مرکز تحقیقات سیستم های هوشمند (CISR) در دانشگاه دیکین ، استرالیا ، شبیه ساز حرکت جهانی (UMS) را که به طور موقت فعال شده است ، توسعه داده است. UMS یک شبیه ساز حرکتی پیشرفته است که بر اساس یک ربات سریال DOF-6 بسیار سفارشی طراحی شده است که دارای یک پاکت حرکت بزرگ ، کنترل سینماتیک با وضوح بالا ، دو محور چرخش مداوم و شتاب واقعی است. این قابلیت را برای مانورهای مختلف معرفی می کند و شبیه سازی حتی غیر معمول ترین حرکت وسیله نقلیه از جمله پاسخ به: شرایط مختلف زمین و آب و هوا ، زاویه های بزرگ شیب ، جابجایی های عمودی بزرگ ، لغزش و واژگونی را امکان پذیر می کند. پلت فرم روبات صنعتی همچنین طول عمر طولانی ، هزینه های کم چرخه عمر و قابلیت اطمینان بالا را تضمین می کند.

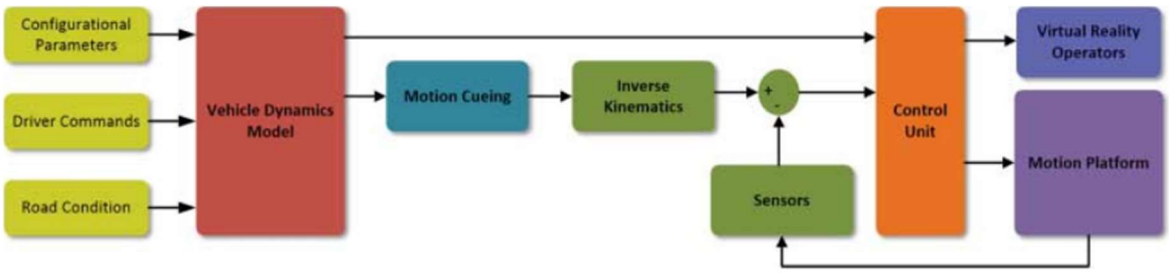
شبیه ساز های حرکت پایین

RVMS سطح پایین در مقایسه با بقیه کم هزینه است و معمولاً شامل یک رایانه ، صفحه نمایش/نمایشگر (های) طرح همراه با یک کابین ساده و صندلی با قابلیت حرکت محدود است. بدیهی است که میزان غوطه وری واقع بینانه در این شبیه سازها در بین سطوح مختلف RVMS کمترین است. با توجه به محدودیت های آنها ، راستی شبیه سازی برای این شبیه سازها عمدتا به کیفیت نشانه های بصری بستگی دارد. فناوری های اخیر در تجسم ، مانند نمایش سه بعدی و کارت های گرافیکی استاندارد خارج از قفسه ، این شبیه سازها را قادر ساخته است تا سطح واقع گرایی را بهبود بخشند. این پیشرفت شبیه ساز را قادر می سازد تا احساس رانندگی واضحی را در

اختیار کاربران این سیستم ها قرار دهد. بسیاری از این شبیه سازها وجود دارد. با این حال ، کاربرد آنها برای تحقیق و شبیه سازی دینامیک خودرو بسیار محدود است و بنابراین آنها از تمرکز این مقاله خارج هستند.

اجزای شبیه ساز حرکت

تحقیقات قابل توجهی در مورد RVMS انجام شده است ، که در آن ویژگی های سواری و هندلینگ وسیله نقلیه از طریق نشانه های مختلف داخل کابین مورد بررسی قرار گرفته است. در این RVMS ، شبیه ساز می تواند نشانه های اینرسی دقیق را به دلیل شتاب زاویه ای و خطی و همچنین اثرات واقعیت مجازی رانندگی ارائه دهد. بنابراین ، چنین RVMS دارای سطح اعتبار و راستی کافی است و می تواند برای آزمایش خودرو مورد استفاده قرار گیرد.



تصویر یک: نمودار بلوک RVMS شامل قطعات اصلی

RVMS شامل چندین جزء به هم پیوسته است ، از جمله مدل دینامیک خودرو ، پلت فرم حرکت ، واحد کنترل ، حرکت حرکت ، حرکت های معکوس و اپراتورهای واقعیت مجازی همانطور که در شکل 1 نشان داده شده است ، راننده نیز در حلقه این بلوک دیاگرام قرار دارد تا وسیله نقلیه مجازی را هدایت کرده و کیفیت سواری و هندلینگ مربوطه را حس کند. برای RVMS که برای اهداف تحقیق استفاده شده است ، نتایج شبیه سازی خودرو باید مورد تجزیه و تحلیل و ارزیابی کیفی و کمی قرار گیرد. در تجزیه و تحلیل کیفی ، راننده باید با تمرکز بر زیر سیستم های درگیر در هر سناریوی رانندگی ، حس سواری و کیفیت رانندگی را تعیین کند.در مقابل ، تجزیه و تحلیل کمی بر اساس پردازش سیگنال برخی از پارامترهای اندازه گیری شده خودرو است. برای این منظور ، معیارها و استانداردهای خاصی وجود دارد که معمولاً برای ارزیابی عملکرد سواری و هندلینگ استفاده می شود.

مدل دینامیک خودرو

مدل پویای خودرو یک برنامه کامپیوتری است که چندین ورودی از پارامترهای خودرو ، دستورات راننده و شرایط جاده ، برخی از آنها از طریق سیستم های جمع آوری داده دریافت می کند و خروجی را به واحد نشانگر حرکت ارائه می دهد. ورودی دستورات راننده شامل شتاب ، ترمز و فرمان است ، در حالی که ویژگی های جاده شامل مسیر ، شیب و مشخصات سطح است. خروجی های مدل پویای خودرو ، پاسخ های پویای خودرو شامل موقعیت ها ، سرعت ها و شتاب های خطی و زاویه ای است.

سکوی حرکت

واحد سکوی حرکت در بلوک دیاگرام شکل 1 مکانیزمی برای تکرار حرکت کابین است. از طریق این واحد ، راننده ای که در یک کابین خلبان کامل یا کامل خودرو قرار می گیرد ، می تواند کیفیت سواری و هندلینگ مصنوعی را احساس کند. این امر با ارائه شتابهای موقتی با درستی بالا برای تحریک درک اینرسی از راننده به دست می آید. برای اهداف شبیه سازی ، سه حرکت در نظر گرفته می شود: حرکت طولی (رانندگی و ترمز) ، حرکت جانبی (هدایت و فرمان) و حرکت عمودی (سیستم تعلیق و میرایی). فرض بر این است که این سه حرکت جدا شده اند. با این حال ، یک مدل کوپل شده دقیق تر نشان داده شده است. پلت فرم حرکت معمولاً دارای فضای کاری

محدود است. بنابراین شتابهای خطی پایدار (افزایش ، تاب و حرکت) عمدتاً با هماهنگی کج صاف و آهسته انجام می شود. این پلت فرم می تواند حرکات زاویه ای (رول ، گام و خم کردن) را با محدودیت کمتری تقلید کند.

ارتعاش فرکانس بالا یک وسیله نقلیه را می توان با استفاده از یک سیستم ارتعاشی که باید به کابین یا قاب صندلی متصل شود ، اجرا کرد. ویپراتور با داده های شبیه سازی شده یا اندازه گیری شده برای مشخصات سطح جاده خاص کنترل می شود. مکانیسم های مختلفی برای بستر حرکت استفاده می شود ، مانند ربات کابلی ، سکوی حرکت کروی ، سکوی حرکتی محور شما ، پلت فرم حرکتی پنج محوره مکانیسم دزدمونا ، مکانیزم استوارت (هگزاپود) و دستکاری کننده سریال.

نشانه حرکتMotion cueing

در شکل 1 ، واحد نشانگر حرکت ورودی را از مدل دینامیک خودرو می گیرد و موقعیت و جهت قابل درک و قابل قبولی را برای سکوی حرکت ایجاد می کند. رویکرد کلاسیک برای نشانه گذاری حرکت از طریق دنباله ای از مقیاس بندی ، فیلترینگ و نگاشت تبدیل بین قاب مختصات خودرو و قاب مختصات سکو انجام می شود. این عملیات لازم است تا امکان تطبیق مسیر برای دو فریم مختصات فراهم شود. حرکت حرکتی باید محدودیت های مفاصل و فضای کار سکوی حرکتی را برای اندازه و فرکانس پارامترهای سینماتیک در نظر بگیرد. کانال هماهنگی شیب در یک الگوریتم علامت حرکت کلاسیک مسئول ایجاد شتابهای پایدار با فرکانس پایین است. از طریق این عملیات ، شتابهای ترجمه ای با ارزش بالا را می توان با استفاده از پارامترهای سینماتیکی زاویه ای بستر حرکت تکرار کرد.

سینماتیک معکوس

بلوک سینماتیک معکوس در شکل 1 مسئول در نظر گرفتن موقعیت ، سرعت و شتاب خودرو در هنگام در نظر گرفتن محدودیت های مربوطه است. الگوریتم سینماتیک معکوس مقادیر متغیرهای مفصل (چرخشی یا منشوری) را برای موقعیت و جهت مورد نیاز کابین ، که از بلوک نشانه حرکت گرفته شده است ، پیدا می کند. تعدادی از چالش ها در سینماتیک معکوس وجود دارد ، از جمله اجرای زمان واقعی الگوریتم برای پلت فرم حرکت ، با توجه به محدودیت های عملی و محدودیت های مکانیسم ، یا بهینه سازی با توجه به ویژگی های منحصر به فرد و فضای کار.

واحد کنترل

واحد کنترل عمدتا وظیفه کنترل سکوی حرکت را بر عهده دارد. شکل 1 نشان می دهد که ورودی واحد کنترل دستوراتی است که توسط بلوک سینماتیک معکوس ارائه شده است. خطاهای بین این فرمان و پارامترهای واقعی سینماتیک پلت فرم توسط واحد کنترل برای محاسبه خروجی کنترل مناسب برای اعمال بر روی محرک ها استفاده می شود. فرمان خروجی این واحد لازم است تا با محدودیت های فیزیکی سکوی حرکتی و احساس انسان مطابقت داشته باشد. یک کنترل کننده مشتق متناسب (PID) یکی از متداول ترین عملکردهای کنترلی است که به دلیل سادگی ، استحکام و قابلیت مناسب برای طیف وسیعی از برنامه ها برای این منظور استفاده می شود. پارامترهای این تابع کنترلی مقادیر افزایش متناسب ، انتگرال و افتراق هستند که به صورت تجربی یا با استفاده از روشهای تنظیم PID تنظیم شده اند.

اپراتور های واقعیت مجازی

در حین شبیه سازی ، راننده نیاز به نشانه های بصری دقیق و کافی برای مانور و هدایت وسیله نقلیه مجازی در شرایط واقعی دارد. افزایش کیفیت درک با افزودن انواع مختلف نشانه های واقعیت مجازی می تواند غوطه وری واقع گرایانه را به طرز چشمگیری افزایش دهد. بنابراین این امر باعث می شود که راننده عملکرد خودرو را درک کند. انواع مختلف واقعیت مجازی نشانه های بصری ، نشانه های شنوایی و بازخورد نیرویی هستند. اپراتورهای واقعیت مجازی در شکل 1 به اطلاعاتی در مورد ویژگی های محیطی ، پارامترهای عملیاتی و بازخورد از مدل دینامیکی خودرو (پارامترهای سینماتیکی خطی و زاویه ای) نیاز دارند. این عملیات بازخورد بصری ، شنیداری

و نیرویی به راننده کمک می کند تا از وضعیت واقعی نزدیک بماند و با زیر سیستم های مختلف خودرو تعامل داشته باشد. چندین مثال از این عملگرها در شبیه سازهای حرکتی موجود یافت می شود. عملیات به طور خلاصه به شرح زیر شرح داده شده است:

نشانه بصری

نشانه شنوایی

بازخورد اجباری

سناریو های تست دینامیک خودرو مبتنی بر شبیه ساز

توانایی RVMS برای شبیه سازی رفتار دینامیک خودرو اجازه می دهد تا از آنها برای ارزیابی و توسعه دینامیک خودرو استفاده کنید. مجموعه ای از روش های آزمون استاندارد وجود دارد. که شامل پویایی خودرو است. این روش های آزمایشی شامل یک سری مانورهای شتاب ، ترمز و چرخش است. اگر شبیه ساز بتواند عملکرد دینامیک خودرو را به طور قابل اعتماد و دقیق نشان دهد ، می توان برای RVMS استفاده کرد. این امر مستلزم عملکرد با عملکرد بالا از اجزای مختلف RVMS از جمله نشانه حرکت ، سینماتیک معکوس ، حلقه کنترل و پلت فرم حرکت است.

بررسی عملکرد و کارایی

تمام حرکتهای خطی و چرخشی خودرو با رفتار جابجایی وسیله نقلیه درگیر است. با این حال ، آزمونهای ارزیابی مربوطه عمدتا به طولی (افزایش) ، جانبی (تاب) و حرکت خمیدگی می پردازد. آگاهی از طیف وسیعی از پارامترها و ویژگی ها برای زیر سیستم های مختلف از جمله تایر ، فرمان ، ترمز ، سیستم تعلیق ، مرکز ثقل و وزن توزیع بر روی محورها برای درک نکردن رفتار جابجایی یک وسیله نقلیه ضروری است. عملکرد وسیله نقلیه جاده ای تحت تأثیر واکنش وسیله نقلیه به فرمان فرمان و ورودی های محیطی مانند اختلالات جاده قرار می گیرد. از این رو عملکرد هندلینگ یک وسیله نقلیه جاده ای به طور مستقیم به کنترل پذیری و پایداری وسیله نقلیه در مانورهای مختلف مانند چرخش یا تغییر خط اشاره دارد.

قابلیت کنترل زاویه فرمان

تعدادی سناریو برای ارزیابی قابلیت کنترل خودرو در حین حرکت دورانی وجود دارد. در این سناریوهای آزمایشی ، عملکرد فرمان خودرو با رفتار فرمان خودرو ارزیابی می شود. رفتار فرمان شامل سرعت انحراف ، مسیر (خمیدگی) ، شتاب جانبی و گشتاور فرمان است. علاوه بر این ، پارامترهای دیگری وجود دارد که می توانند برای ارزیابی بهتر در نظر گرفته شوند ، مانند خطی بودن ، دقت ، تلاش ، رفتار روی مرکز و قابلیت بازگشت سیستم فرمان.

طیف وسیعی از مانورها در سناریوهای آزمایشی در نظر گرفته شده است. موسسات و شرکتهای تحقیقاتی مختلف مانورها و روشهای آزمایشی خود را دارند. با این حال ، از دیدگاه کلی ، مانورهای تغییر مسیر و تغییر مسیر را می توان در شعاع ثابت ، سرعت ثابت و چرخش ثابت زاویه فرمان گروه بندی کرد. در حال حاضر ، کنترل فرمان روباتیک برای اجرای دقیق مانورهای آزمایشی استفاده می شود. قابل توجه است که RVMS می تواند مانورها را با دقت بیشتری تقلید کند زیرا امکان استفاده از سناریوهای از پیش تعریف شده در شبیه ساز حرکت وجود دارد.

ثبات جانبی و انحراف

پایداری خودرو یک مسئله مهم برای کیفیت جابجایی و ایمنی مسافران است. عدم پایداری می تواند باعث چرخش خودرو یا منحرف شدن از جهت مورد نظر در طول پیچ شود. چنین بی ثباتی ای ناشی از لغزش و افزایش رفتار بیش از حد لاستیک ها است. سناریوهای مانور ترکیبی برای ارزیابی پایداری جانبی یا خمیدگی وسایل نقلیه استفاده می شود. به عنوان مثال ، مانور کامل ترمز در هنگام چرخش با حداکثر زاویه فرمان باید منجر به سرعت پایدار جانبی و خم شدن خودرو شود. آگاهی از ورودی های راننده ، سرعت جلو ، سرعت انحراف ، شتاب جانبی ، زاویه لغزش واقعی ، و انتقال بار محور ، با توجه به سختی سیستم تعلیق ، برای نظارت و بهبود پایداری خودرو ضروری است. از سیستم VSC برای تضمین پایداری خودرو استفاده می شود. کنترل کننده VSC تفاوت بین حرکت برآورد شده

و واقعی وسیله نقلیه را در طول چرخش یا انحراف کاهش می دهد. این امر با اعمال ترمز چرخ جداگانه یا کنترل بر روی توزیع گشتاور پیشرانه به دست می آید. مکانیزم ضد رول یا نوار تثبیت کننده یکی دیگر از راه حل های امیدوار کننده است که در مدل دینامیک خودرو برای بهبود رفتار چرخش خودرو استفاده شده است.

کیفیت رانندگی

کیفیت رانندگی عمدتاً تحت تأثیر ارتعاشات عمودی چند هارمونیکی (حداکثر 100 هرتز) است که در داخل خودرو احساس می شود. تجزیه و تحلیل جامع و شبیه سازی عملکرد سواری مستلزم در نظر گرفتن ویژگی های سطح جاده ، ارتعاش وسیله نقلیه و واکنش انسان به ارتعاش است. بر اساس چنین تحلیلی ، زیر سیستم های دینامیک خودرو شامل تایر ، سیستم تعلیق یا صندلی را می توان برای کاهش سطح ارتعاش اصلاح یا بهینه سازی کرد. استانداردهای متعددی وجود دارد که محدودیت ارتعاش را برای راحتی مسافران تعیین می کند. برجسته ترین استاندارد در این زمینه ISO 2631 است که علاوه بر ارتعاش را در نظر می گیرد سطح ، زمان قرار گرفتن در معرض ارتعاش. این امر به این دلیل است که تفاوت های قابل توجهی در محدوده راحتی مسافر برای زمان های مختلف قرار گرفتن در معرض ارتعاش وجود دارد.

قابلیت کشش

آزمایش عملکرد خودرو در حرکات طولی برای طیف وسیعی از کاربردها بسیار مهم است. زیر سیستم های مختلف خودرو از جمله پیشرانه (موتور ، گیربکس و دیفرانسیل) ، تایر و طراحی آیرودینامیک در آزمایش های کشش تأثیرگذار هستند. در طول این آزمایشها ، حداکثر سرعت ، زمان شتاب و تعامل بین تایر و جاده با سناریوهای مختلف مانند مانورهای حلقه بسته ارزیابی می شود. سپس نتایج مورد بررسی قرار می گیرد تا مشخص شود که آیا در هنگام شتاب شدید ، لغزش تایرها یا از دست دادن تماس جاده ای که باعث از بین رفتن کشش می شود رخ داده است یا خیر. به منظور دستیابی به کنترل بهتر بر کشش ، سیستم های کنترل کشش (TCS) از طریق ترمز لاستیکی فردی ، کنترل دریچه گاز یا قفل دیفرانسیل استفاده می شود. کاربرد یک آزمایش مبتنی بر شبیه ساز برای وسایل نقلیه به دلیل محدود بودن فضای کار سکوی حرکتی و این واقعیت است که حرکت طولی عمدتا توسط نشانه های بصری و نه نشانه های حرکت تقلید می شود.

عملکرد ترمزگیری

زمان و فاصله برای استراحت کامل به دلیل ترمز کامل ، آزمایشهای مهمی برای ارزیابی پویایی خودرو و سیستمهای ایمنی فعال تعبیه شده مانند ترمز ضد قفل (ABS) است. استانداردهای مربوط به این آزمایش محدودیت های مجاز زمان و مسافت را برای سرعت معین ، مشخصات سطح جاده و ارتفاع نشان می دهد. مدلهای دقیق و غیر خطی تایرها ، سیستم ترمز و اصطکاک جاده برای استفاده از RVMS برای ارزیابی عملکرد ترمز مورد نیاز است. این امر ارزیابی را تا حدی دشوار و از نظر فیزیکی غیرقابل اعتماد می کند.

شبیه سازی

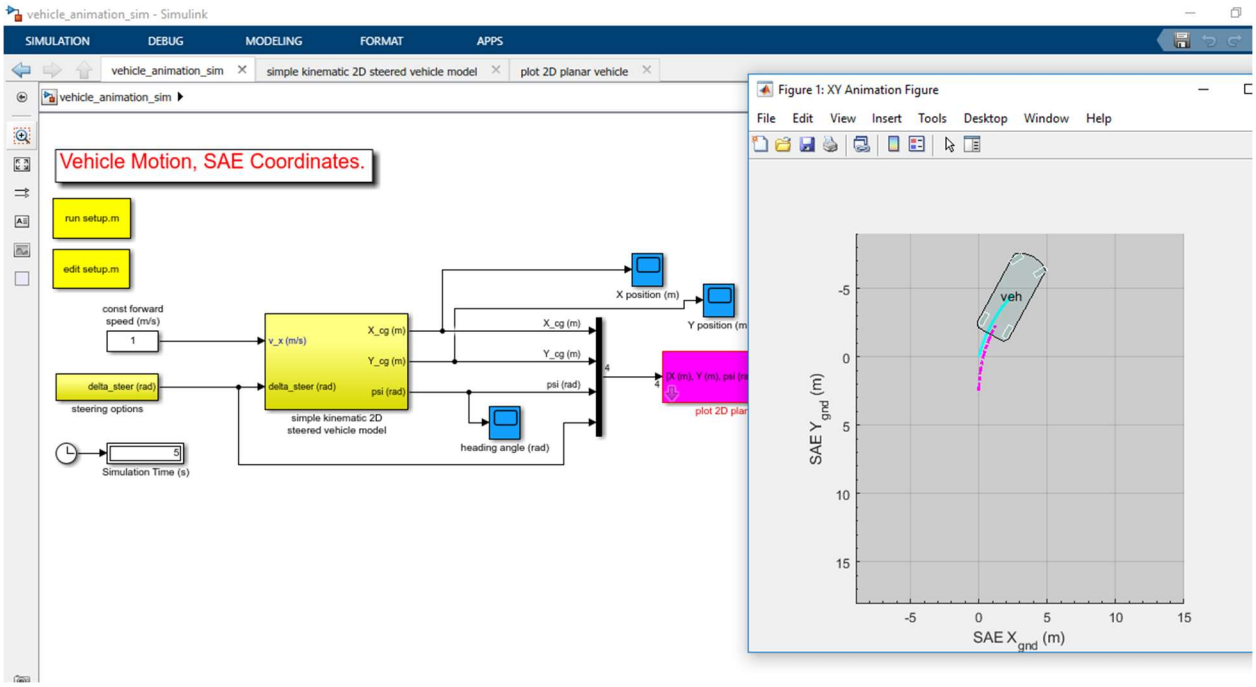
این مدل Simulink برای موقعیت XY در فزیم ثابت زمین، حرکت ساده خودرو را شبیه سازی می کند و نتیجه را با استفاده از Matlab نمایش می دهد. فرمان آکرمن اشاره به پیکربندی هندسی دارد که به هر دو چرخ جلو اجازه می دهد با زاویه مناسب هدایت شوند تا از لغزش تایر جلوگیری شود. ... تفاوت زاویه هدایت چرخ جلو به عنوان تابعی از زاویه فرمان ورودی به انگشت پویا معروف است. رابطه فرمان Ackermann delta_Ack = L / R تقریبی برای وسیله نقلیه با طول فاصله بین دو محور L و عرض مسیر W است که بدون لغزش بر روی یک دایره شعاع ثابت با سرعت کم می چرخد. مختصات SAE ثابت با بدنه از +x به جلو خودرو اشاره می کند ، +y را از پنجره سمت سرنشین ، و +z را به پایین ، به داخل زمین نشان می دهد. زمینه مختصات SAE XY دارای Z به سمت پایین به سمت زمین است.

این مدل با استفاده از Matlab R2020 توسعه و آزمایش شده است و باید با اکثر نسخه های دیگر کار کند. تابع s-animation یک نسخه اصلاح شده از تابع s-m سطح 1 از مثال Mathworks است که در sanim.m ارائه شده است. نوشتن فرمت های متحرک

.jpeg بر روی دیسک در هر فاصله انیمیشن ، شبیه سازی را به میزان قابل توجهی کند می کند. فریم های متحرک را در ثانیه ، anim_fps ، در setup.m تنظیم کنید و دوباره setup.m را اجرا کنید. فراموش نکنید که خدمات همگام سازی فایل خود را خاموش کنید تا از همگام سازی همه فایل های تصویری جدید جلوگیری کنید. با فشردن فایل .zip ، تغییر دایرکتوری ها به پوشه ، و سپس اجرای setup.m در خط فرمان Matlb شروع به کار کنید. با این کار فضای کار پاک می شود و سپس متغیرهای لازم را برای اجرای مدل Simulink و انیمیشن مرتبط با آن پر می کند. همچنین فایل مدل Simulink را باز می کند. Play یا Simulation را فشار دهید برای اجرای مدل Simulink اجرا کنید. بلوک تابع آبی روشن پنجره شکل متحرک دو بعدی را نشان می دهد و وسیله نقلیه فرمان را با فریم های anim_fps در ثانیه نمایش می دهد.

اجرا

در این مدل سه زیر سیستم وجود دارد، یکی برای تغییر زاویه فرمان و دیگری برای تغییر جهت به چپ و راست و در آخر برای تبدیل مسیر و جهت حرکت وسیله نقلیه به انیمیشن متحرک.



تصویر یک: مدل شبیه سازی شده

روش گرادیان مزدوج

روش گرادیان مزدوج یا روش گرادیان همیوغ در ریاضیات، الگوریتمی برای حل سیستم معادلات خطی می‌باشد. معادلاتی که ماتریس آنها متقارن و مثبت معین است. این روش از نوع الگوریتم‌های تکراری می‌باشد. بخش مهمي از مسائل قابل هدايت در مهندسي، از جمله مهندسي رباتيک، مسائل کنترلي از نوع تنظيم کننده هستند. از طرفي روش هاي گرادیان مزدوج تعميم يافته و روش نشاندن قابليت هاي توانمندي در حل اين مسائل دارند. همچنين چگونگي کاربرد آنها در محاسبه مسير و کنترل بهينه يك وسيله نقليه مورد استفاده قرار می گیرد. عملکرد گرادیان مزدوج برای بهینه سازی کنترل دینامیک خودرو است. متغیرهای کنترل بهینه عبارتند از انتقال گشتاور دیفرانسیل عقب فعال و زاویه فرمان فرمان عقب فعال ، در حالی که وظایف بهینه سازی عبارتند از ردیابی مسیر و به حداقل رساندن رول برای مانور تغییر خط دوگانه. با اینکه روش نیوتن یکی ازموثرترین روش ها برای حل مسائل بهینه سازی نامقید می باشد، اما این روش دارای معایبی است که باعث کندی روند می شود.

1. هزینه بالای محاسبه ی ماتریس G_k, G_k^{-1} .

2. امکان وارون ناپذیری ماتریس G_k و در نتیجه عدم توانایی در محاسبه قدم نیوتن.

3. کاهشی نبودن الگوریتم در صورتی که G_kمعین مثبت نباشد.

روش های گرادیان مزدوج

روش‌های گرادیان مزدوج از دیرباز به‌عنوان یک خانواده‌ی بسیار مهم از روش‌ها برای حل سیستم‌های معادالت خطی و مسائل برنامه‌ریزی غیرخطی در مقیاس بزرگ مورد استفاده قرار گرفته‌اند. مبحث الگوریتم‌های جهت‌های مزدوج مبحثی سرشار از خالقیت در عرصه‌ی برنامه‌ریزی غیرخطی است که نشان می دهد که تحلیل تفصیلی مسئله‌ی درجه‌ی دوم می‌تواند منجر به پیشرفت‌های عملی قابل ملاحظه‌ای گردد. روش‌گرادیان مزدوج یک روش جهت‌های مزدوج با یک ویژگی خاص می‌باشد وآن این است که در تولید بردارهای جهت، بردار جدید dk را تنها با استفاده از بردار قبلی$dk-1$ محاسبه می‌کند و با دانستن تمام بردارهای قبلی ، $dk-2$، $d1$ $d0$از مجموعه‌ی مزدوج نیازی ندارد. فرمول تکراری یک روش گرادیان مزدوج به صورت زیر است:

$$x_{k+1} = x_k + \alpha_k d_k$$

$$d_k = \begin{cases} -g_k & k = 0 \\ -g_k + \beta_k d_{k-1} & k \geq 1 \end{cases}$$

الگوریتم۱ (روش‌های گرادیان مزدوج)

ورودی: بردار آغازین $x_0 \in R^n$.

گام (۰): قرار بده $k = 0$ و $d_0 = -g_0 = b - Qx_0$ را محاسبه کن.

گام (۱): تا زمانی که $g_k \neq 0$، مراحل زیر را دنبال کن.

گام (۲): α_k را از رابطه‌ی زیر محاسبه کن

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}.$$

گام (۳): قرار بده:

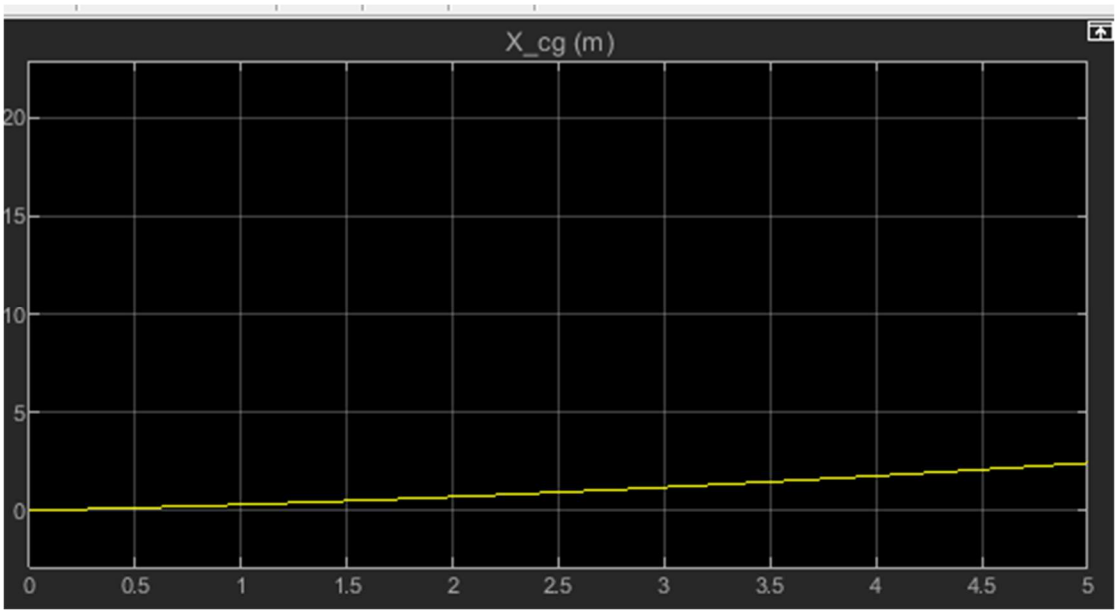
الف) $x_{k+1} = x_k + \alpha_k d_k$.

ب) $g_{k+1} = Qx_{k+1} - b$.

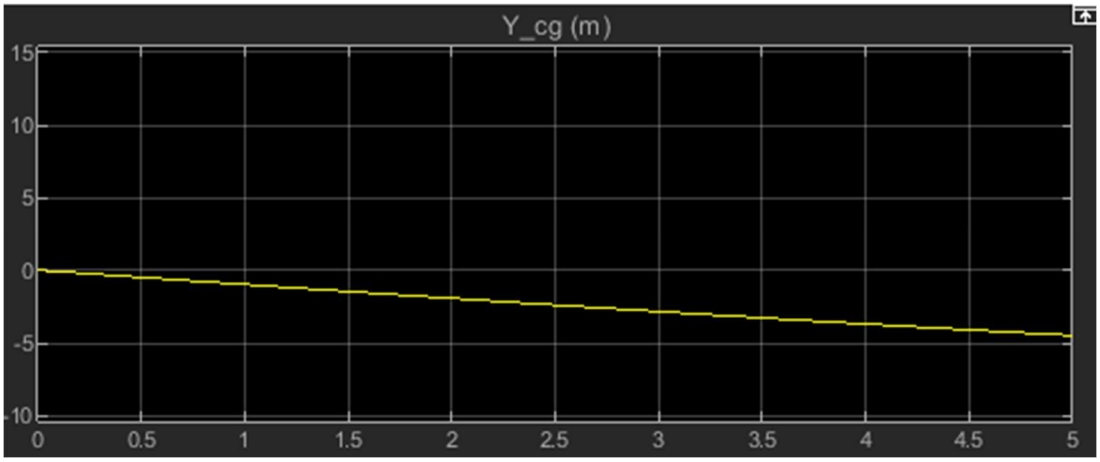
ج) $\beta_{k+1} = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$.

گام (٤): جهت جستجو را از رابطه‌ی $d_{k+1} = -g_{k+1} + \beta_{k+1} d_k$ محاسبه کن.

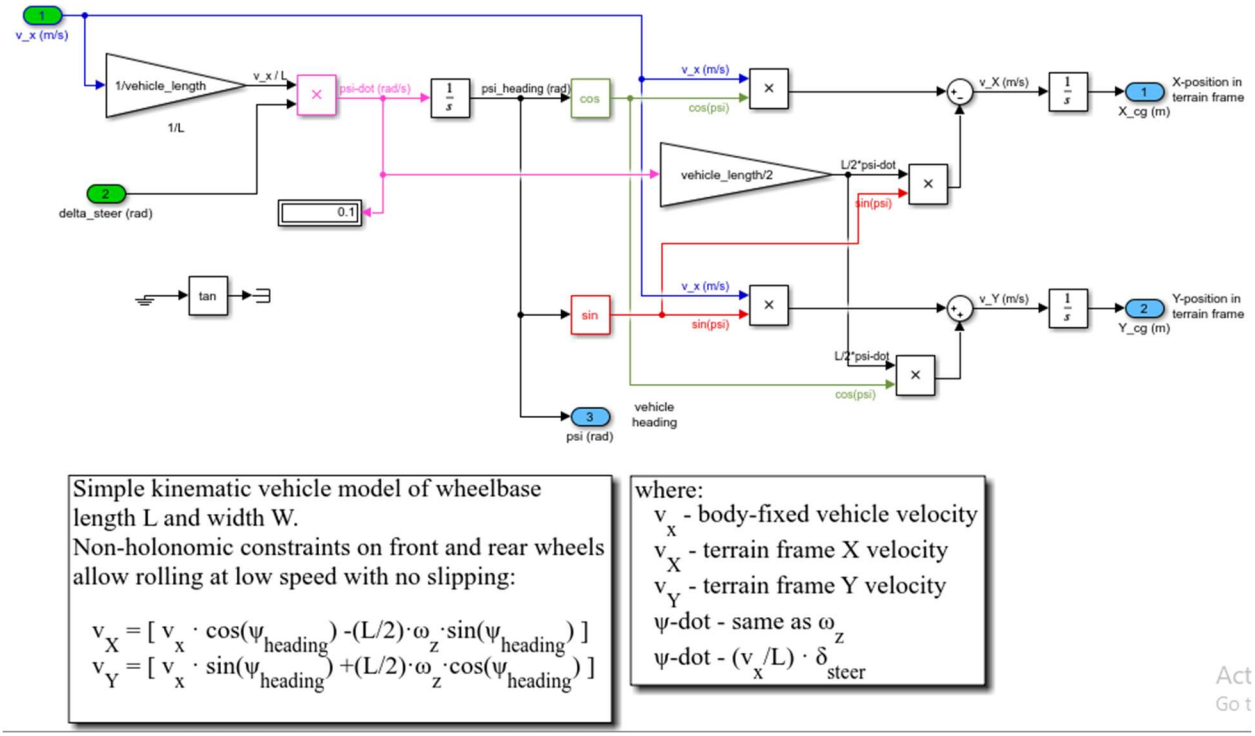
گام(۵): قرار بده $k = k + 1$ و به گام(۱) برو.



تصویر دو: X_cg Conjugated Gradients گرادیانهای مزدوج



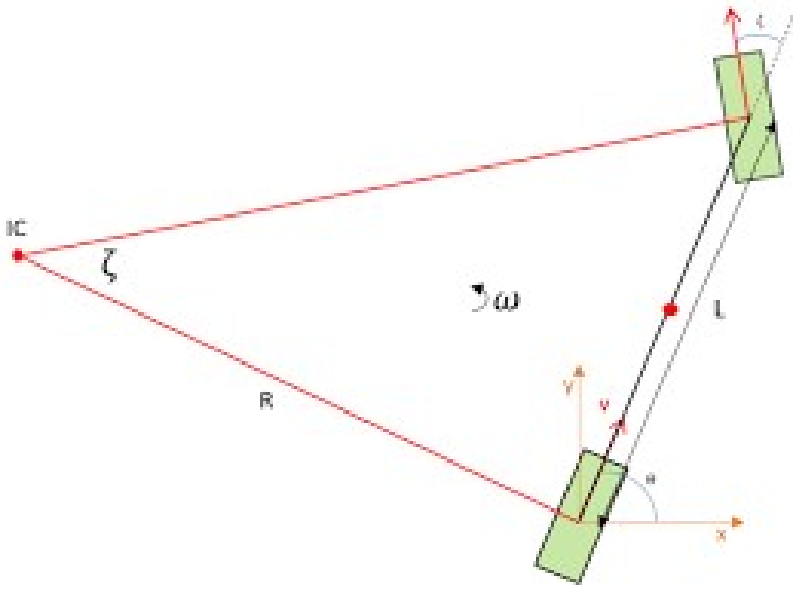
تصویر سه: Y_cg Conjugated Gradients گرادیانهای مزدوج



تصویر چهار: مدل فرمان حرکتی دو بعدی

مدل سینماتیک

طبق تعریف ، سینماتیک شاخه ای از مکانیک است که به حرکت اجسام بدون اشاره به نیروهایی که باعث حرکت می شوند توجه دارد. در زمینه سینماتیک چرخ ، این به کامبر و پنجه اشاره می کند ، که دو پارامتر اصلی دینامیک خودرو ما هستند.Camber و Toe با هم در دینامیک خودرو موثر هستند. در حالیکه مدل دوچرخه از یک مدل 4 چرخ استفاده می کند و چرخ های جلو و عقب را به ترتیب با هم ترکیب می کند تا یک مدل دوچرخ را تشکیل دهد (از این رو نام مدل دوچرخه). از اینرو در مدل دوچرخه به جای برخورد با 4 چرخ و 2 زاویه فرمان ، ما فقط باید 2 چرخ و 1 زاویه فرمان را در نظر بگیریم.



تصویر پنج: مدل دینامیکی دوچرخه

ورودی های ما به مدل سرعت خودرو v و زاویه فرمان است. ما از مدل خود برای تعیین سرعت x & v و همچنین سرعت زاویه ای استفاده می کنیم. با استفاده از این نرخ تغییرات می توانیم موقعیت x & y و همچنین عنوان خودرو را محاسبه کنیم.

مرکز لحظه ای چرخش

اگر وسیله نقلیه را در برهه ای از زمان نگاهداریم، می توانیم خطی عمود بر بردار سرعت در هر نقطه از وسیله نقلیه ترسیم کنیم. ما می توانیم این خط را در محور جلو ، محور عقب ، مرکز ثقل یا هر نقطه دیگری از وسیله نقلیه ترسیم کنیم (3 نقطه فوق الذکر به طور کلی 3 نقطه مفید برای انتخاب در نظر گرفته می شوند). جایی که همه این خطوط به هم می رسند به عنوان مرکز لحظه ای چرخش شناخته می شود. در آن لحظه در زمان ، نقطه ای است که وسیله نقلیه در اطراف آن می چرخد. در شکل پنج، ما از 3 نقطه ای که در بالا ذکر شد 3 خط عمود کشیده ایم. به یاد داشته باشید ما شرایط بدون لغزش را فرض می کنیم ، بنابراین خط چرخ عقب عمود بر جهت چرخ عقب که در جهت وسیله نقلیه حرکت می کند کشیده شده است. خط چرخ جلو عمود بر چرخ جلو کشیده می شود که در زاویه فرمان خاصی قرار دارد و خط در مرکز ثقل عمود بر سرعت آن که زاویه ای نسبت به جهت حرکت وسیله نقلیه دارد کشیده می شود. نقطه ای که این خطوط به هم می رسند به عنوان مرکز لحظه ای چرخش شناخته می شود.

دلیل اینکه ما می توانیم دو چرخ جلو و عقب را با هم ترکیب کنیم به دلیل این مفهوم مرکز لحظه ای است. ما دو زاویه مختلف فرمان را با هم ترکیب کرده و یک زاویه فرمان واحد را تشکیل می دهیم. جنبه کلیدی قابل توجه این است که زاویه فرمان جدید همچنان باعث می شود که وسیله نقلیه در همان مرکز لحظه ای بچرخد ، همانطور که در هر چهار چرخ کار می کرد. این بدان معناست که معادلات حرکتی که در مدل 4 چرخ اعمال می شود همچنان در مدل دوچرخه دو چرخ اعمال می شود ، اما به دلیل سادگی مدل ، بدست آوردن آن آسان تر خواهد بود.

استخراج معادلات حالت وسیله نقلیه

ما می توانیم از این مرکز لحظه ای چرخش برای بدست آوردن معادلات حالت که برای توصیف حرکت وسیله نقلیه استفاده می کنیم و در نهایت خروجی های مورد نظر خود را (حالت خودرو) محاسبه کنیم ، استفاده کنیم. بسته به سناریو ، بدست آوردن معادلات در نقاط مرجع مختلف در اطراف وسیله نقلیه مفید خواهد بود. 3 نقطه مهم عبارتند از: محور جلو ، محور عقب و مرکز ثقل.

نقطه مرجع محور عقب

بیایید با قرار دادن نقطه مرجع خود در محور عقب خودرو شروع کنیم. از آنجا که ما نقطه مرجع را در محور عقب قرار داده ایم ، سرعت خودرو در راستای جهت چرخ عقب در زاویه محور x عمل می کند.

بنابراین ما می توانیم سرعت را به اجزای x & y تقسیم کنیم که منجر به موارد زیر می شود:

$$\dot{x}_r = v * cos(\theta)$$

$$\dot{y}_r = v * sin(\theta)$$

معادله یک: سرعت در راستای x,y

برای یافتن شتاب زاویه ای می توانیم از فرمول زیر استفاده کنیم:

$$\dot{\theta} = \omega = \frac{v}{R}$$

معادله دو: شتاب زاویه ای

جاییکه R فاصله عمود از نقطه مرجع تا IC است. اکنون ما باید R را از نظر ورودی فرمان خود استخراج کنیم. با استفاده از هندسه زاویه اصلی ، می بینیم که زاویه متقاطع بین خطوط عمود بر جلو و عقب نیز وجود دارد. از این پس ما می توانیم بنویسیم:

$$tan(\zeta) = \frac{L}{R} \text{ so } R = \frac{L}{tan(\zeta)}$$

معادله سه: زاویه متقاطع

با جایگزینی این معادله اصلی ما می توانیم 3 معادله حالت زیر را برای نقطه مرجع محور عقب ایجاد کنیم:

$$\dot{x}_r = v * cos(\theta)$$

$$\dot{y}_r = v * sin(\theta)$$

$$\dot{\theta} = \omega = v * \frac{tan(\zeta)}{L}$$

معادله چهار: معادلات سرعت و شتاب زاویه ای محور عقب

مرجع اکسل جلو

برای نقطه مرجع محور جلو ، مشتق مشابهی می توان انجام داد. بسیار شبیه به نقطه مرجع محور عقب است. به خاطر داشته باشید که در حالتی که سرعت خود را در اجزای x & y تعیین می کنید ، زاویه فرمان نیز باید در نظر گرفته شود.

معادلات بدست آمده به شرح زیر است:

$$\dot{x}_f = v * cos(\theta + \zeta)$$

$$\dot{y}_f = v * sin(\theta + \zeta)$$

$$\dot{\theta} = \omega = v * \frac{sin(\zeta)}{L}$$

معادله پنج: معادلات سرعت و شتاب زاویه ای محور جلو

مرکز نقطه مرجع جرم

احتمالاً مهمترین نقطه از 3 نقطه مرجع مرکز مرجع جرم است. از آنجا که ما فرض می کنیم وسیله نقلیه دارای جرم متراکم است ، همه اینها مرکز مرکز جرم است ، برای ما مهم است که بتوانیم مجموعه معادلاتی را که حرکت را در این نقطه توصیف می کنند ، بدست آوریم. این همچنین به ما امکان می دهد زاویه لغزش خودرو یعنی زاویه بین مسیری که وسیله نقلیه در آن حرکت می کند را در مقابل سرعت در مرکز جرم پیدا کنیم. استخراج معادلات برای این نقطه مرجع کمی پیچیده تر از نقاط مرجع جلو و عقب است.

حل سرعت به x & y به روشی مشابه قبلی بدست می آید.

$$\dot{x}_c = v * cos(\theta + \beta)$$

$$\dot{y}_c = v * sin(\theta + \beta)$$

معادله شش: معادلات سرعت و شتاب زاویه ای مرکز ثقل

با این حال ، برای پیدا کردن زاویه فرمان ، ما باید مرحله اضافی پیدا کردن فاصله S را قبل از اینکه بتوانیم شعاع چرخش R را پیدا کنیم ، انجام دهیم.

$$S = \frac{L}{tan(\zeta)}$$

$$R = \frac{S}{cos(\beta)}$$

$$R = \frac{L}{tan(\zeta)cos(\beta)}$$

So therefore:

$$\dot{\theta} = \omega = v * \frac{tan(\zeta)cos(\beta)}{L}$$

معادله هفت: معادلات زاویه فرمان و شعاع چرخش R

با دانستن فاصله جلو محور عقب تا مرکز جرم ، می توانیم زاویه فرمان را به شرح زیر توصیف کنیم:

$$\beta = tan^{-1}(\frac{tan(\zeta)}{L})$$

معادله هشت: معادله زاویه فرمان

بنابراین ، مجموعه معادلات ما برای مرکز نقطه مرجع جرم عبارتند از:

$$\dot{x}_c = v * cos(\theta + \beta)$$

$$\dot{y}_c = v * sin(\theta + \beta)$$

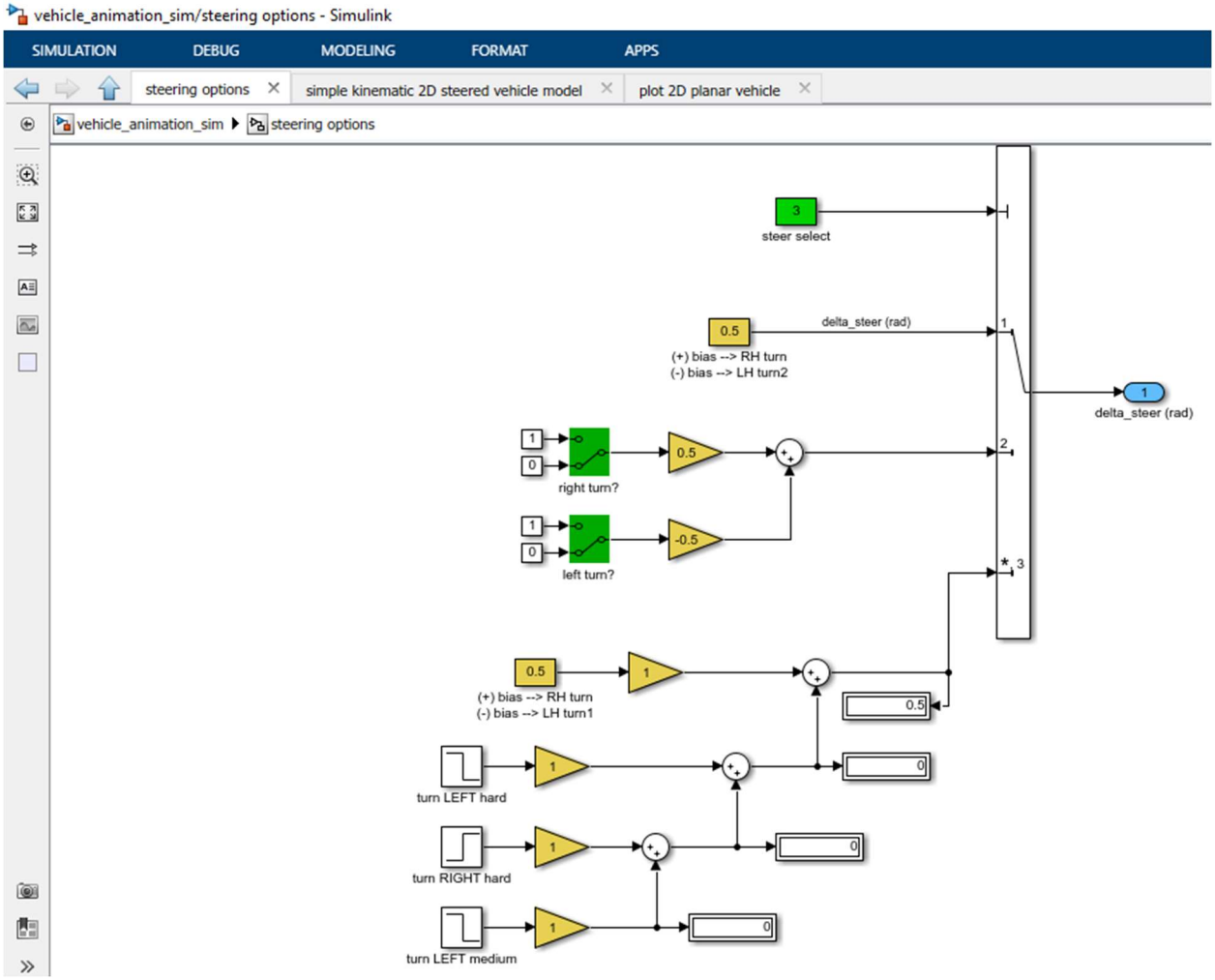
$$\dot{\theta} = \omega = v * \frac{tan(\zeta)cos(\beta)}{L}$$

$$\beta = tan^{-1}(l_r\frac{tan(\zeta)}{L})$$

معادله نه: معادلات نهایی مرکز ثقل

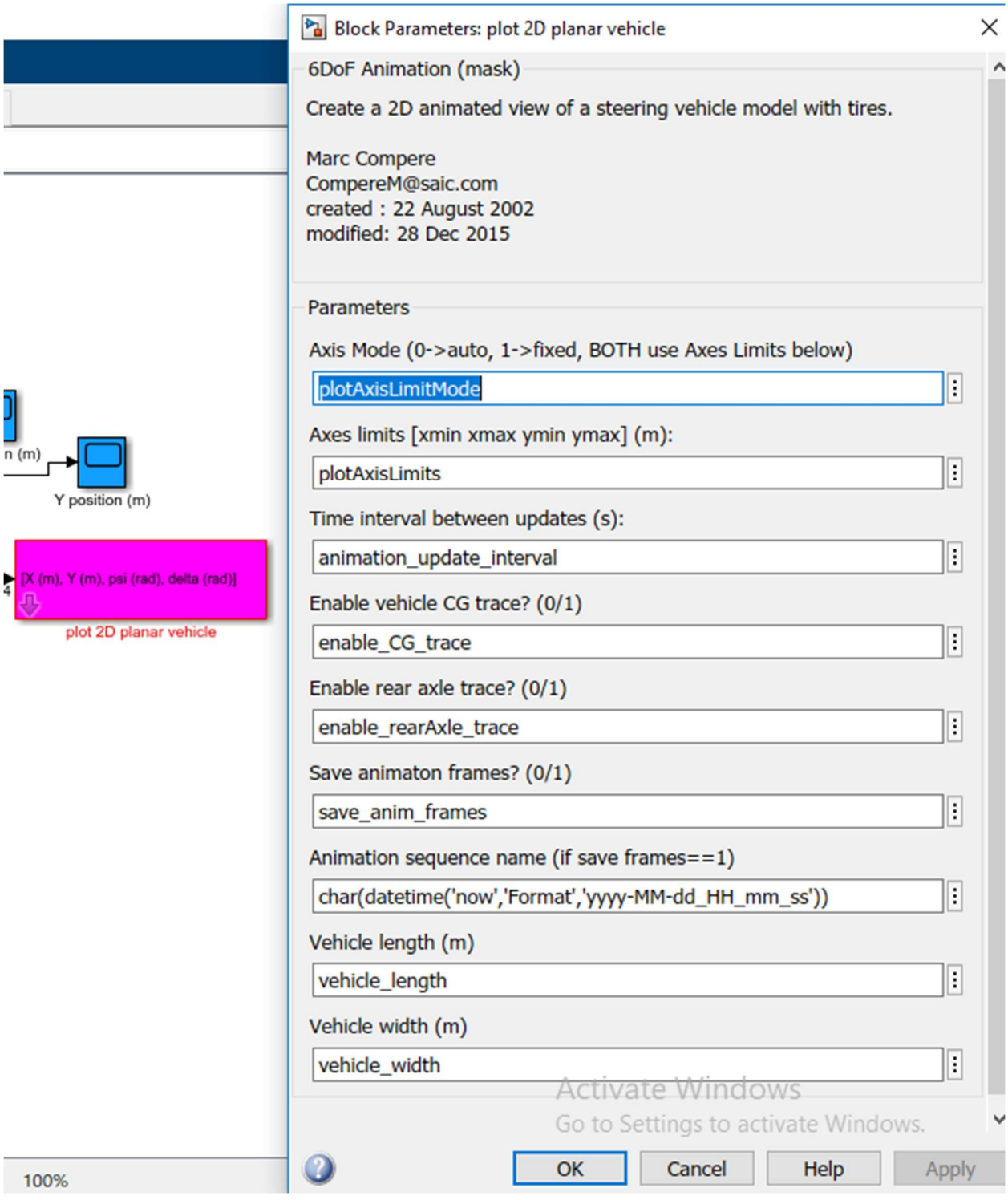
سیستم کروز در ماشین عمل تثبیت سرعت را انجام می دهد و طراحی بخش کنترلی آن به گونه ای است که پاسخ سریع باشد و اثر اغتشاشات در آن کم باشد بسیار مهم است. زمانی که این ابزار را فعال می‌کنید و سرعت خودرو روی عدد خاصی ثابت شده است، این سرعت، در مسیرهای سربالایی، سراشیبی، یا هنگام وزش باد مخالف، تغییر نخواهد کرد. گفتنی است، کروز کنترل این امکان را نیز در اختیار راننده قرار می‌دهد تا بی آن‌که نیازی به فشردن پدال‌های ترمز و گاز باشد، سرعت خودرو را با هر بار فشردن دکمه کنار فرمان، به میزان یک کیلومتر بر ساعت، افزایش یا کاهش دهد.

ما از سرعت و میزان تغییر زاویه فرمان به عنوان ورودی استفاده می کنیم و پس از محاسبه خروجی ها با استفاده از معادلات حالت خود ، قادر به تعیین موقعیت x ، موقعیت y ، جهت خودرو و زاویه فرمان هستیم. ما این معادلات حالت را با استفاده از 3 نقطه مرجع بر روی وسیله بدست آوردیم. یعنی ، محور جلو ، محور عقب و مرکز جرم. هر نقطه مرجع مجموعه متفاوتی از معادلات حالت را به ما می دهد که مهمترین آنها مرکز جرم است که به ما امکان می دهد زاویه لغزش کناری وسیله نقلیه را پیدا کنیم. در حالی که مدل دوچرخه سینماتیک یک مدل وسیله نقلیه بسیار اساسی است با فرض های بسیاری ، هنوز یک بلوک اساسی در درک و ساخت یک مدل خودرو جامع تر است. از اینجا می توانید مفروضات را از مدل حذف کنید و تجزیه و تحلیل دقیق تری از دینامیک خودرو را شامل شوید.

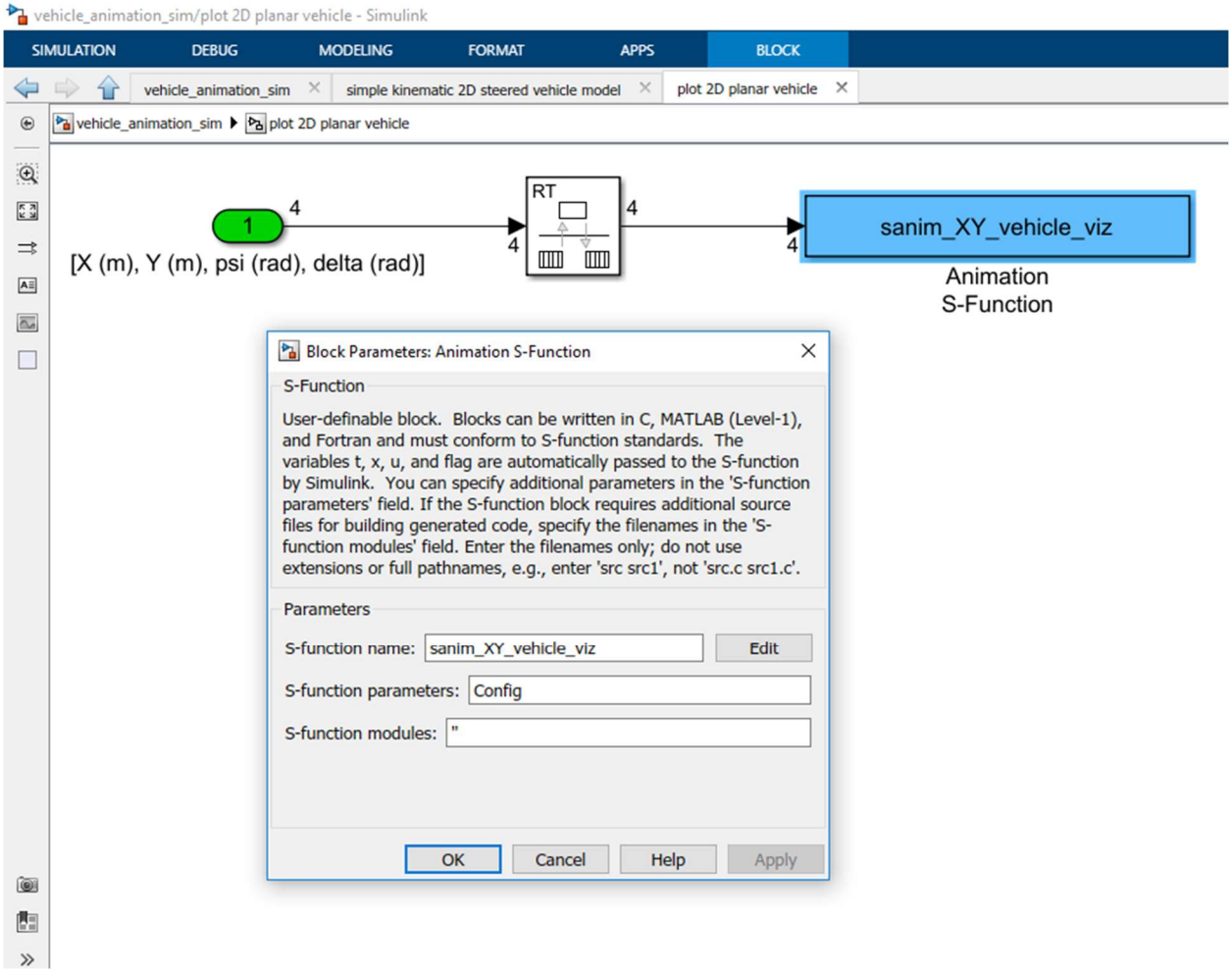


تصویر شش: آپشنهای فرمان – جهت های حرکتی چپ و راست

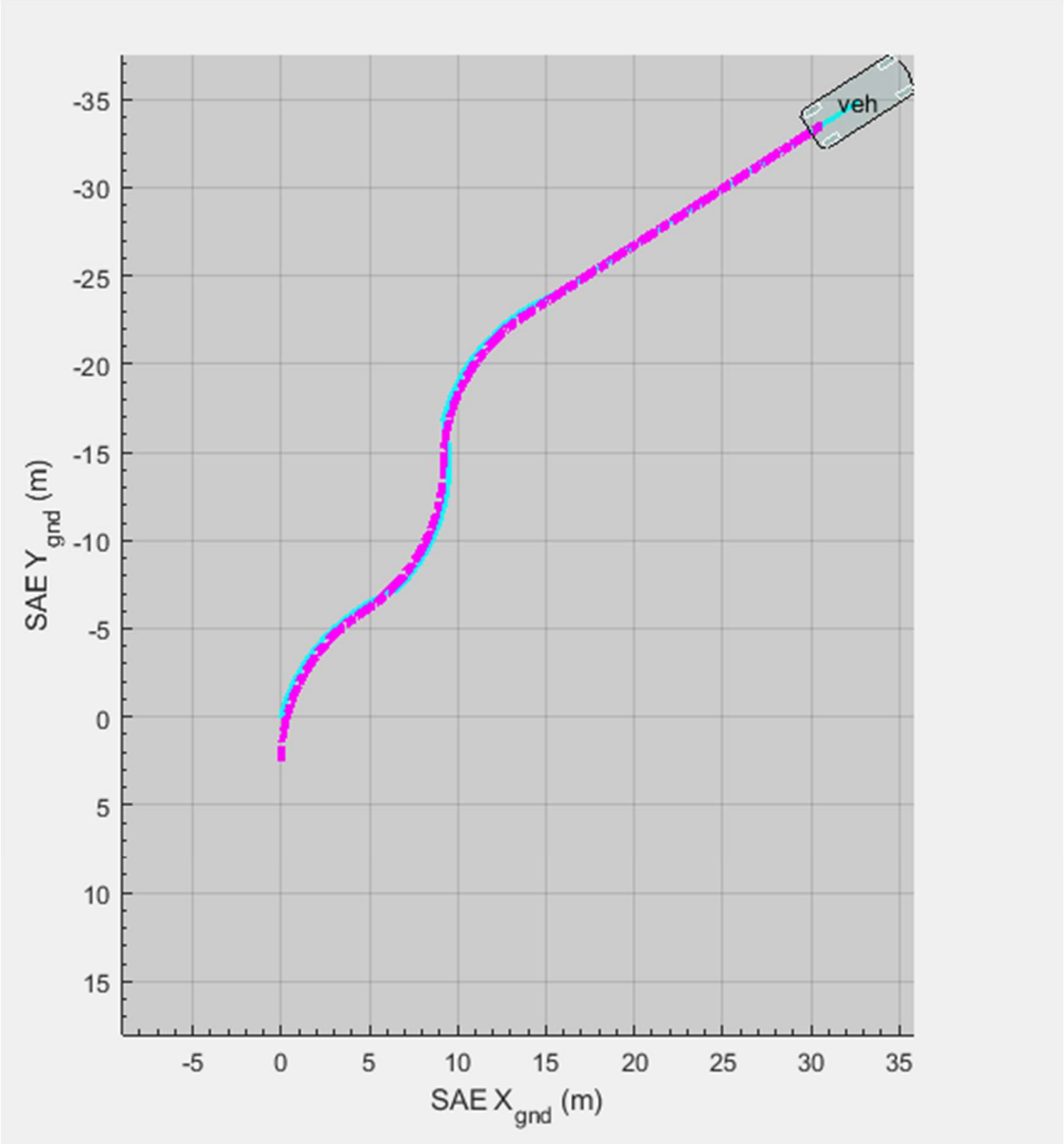
در تصویر شش، در زیر سیستم آپشنهای فرمان، زاویه حرکت فرمان بدست می آید. با استفاده از بلاکهای Gain, Step, Constant Value, Sum زاویه حرکت و همچنین جهت حرکت به سمت راست یا چپ را بعنوان خروجی به سیستم اصلی منتقل می کند.



تصویر هفت: مدل شبیه سازی شده با 6 درجه آزادی



تصویر هشت: شبیه سازی مدل به انیمیشن متحرک



تصویر نهم: شبیه سازی مدل وسیله نقلیه در مدت زمان 50 ثانیه

کدنویسی

[setup.m]

ابتدا این را اجرا کنید ، Simulink ظاهر می شود ، سپس play را فشار دهید تا خودرو شبیه سازی شود

```
clear; clear all ; clear functions

% ----------------------------------------------
% -------  swarm and control parameters  -------
% ----------------------------------------------
```

```
vehicle_length = 5; % (m)
%vehicle_length = 0.1; % (m)
vehicle_width  = 2; % (m)


X_ic = 0*[4*(rand-0.5)+3]; % (m) random ICs on X position, note: rand() is on
[0 1]
Y_ic = 0*[5*(rand-0.5)+0]; % (m) random ICs on Y position
%yaw_ic = 2*pi*(rand(nAgents,1)-0.5); % (rad) random ICs for yaw, or heading
yaw_ic = 1*(-pi/2); % (rad) random ICs for yaw, or heading


% ------------------------------------------
% -----  solver and animation parameters  -----
% ------------------------------------------
h_fixed = 0.05; % (s) fixed solver simulation stepsize

plotAxisLimitMode = 0; % 0->auto, 1->fixed, use Axes Limits in
'plotAxisLimits'
%plotAxisLimits = [-3 5 -3 6]*20; % [xmin xmax ymin ymax]
plotAxisLimits = [-3 5 -3 6]*3; % [xmin xmax ymin ymax]

%anim_fps=20; % (animation frames / second)
anim_fps=10; % (animation frames / second)
%anim_fps=2; % (animation frames / second)
enable_CG_trace=1;       % (0/1) plot animation trace from vehicle CG, or
geometric center
enable_rearAxle_trace=1; % (0/1) enable animation trace from rear axle
                    %       when writing a .jpeg image to file at each
animation interval.
                    %       see writeVideo() at this link for converting into
.avi movies:
                    %
http://www.mathworks.com/help/matlab/examples/convert-between-image-
sequences-and-video.html)

save_anim_frames=0; % (0/1) save animation frames? this slows the simulation
considerably

% animation update rate assuming Simulink clock advances very nearly the wall
clock
C = round(1/(anim_fps*h_fixed)); % see notes below
animation_update_interval=C*h_fixed; % (s) This parameter controls three
things:
                                                %     (1) The animation update
to the screen is updated at this
                                                %         rate which needs to
be an integer multiple of fixed stepsize integration.
                                                %     (2) A frame is saved on
this interval during movie makin (i.e. when movie_parm.save_frames=1)
                                                %     (3) how close to (or how
much faster than) real-time the simulation runs.
                                                %         See timing_notes.txt
for more information on run speeds.
```

```
%radius = abs( vehicle_length / (-0.5) ); % (m) turn radius = L / R
%circumference = 2*pi*radius; % (m) distance traveled around the circle, once
%note: at v_x = 1(m/s), time for 1 revolution is circumference, or t=dist/vel

% bring up the simulink model
vehicle_animation_sim
```

[createAviMovieFromAnimationSequence.m]

اسکریپت m-file کمکی برای تبدیل دنباله ای از تصاویر .jpg به Avi با استفاده از تابع Matlab () VideoWriter.

```
% createAviMovieFromAnimationSequence.m
%
% This script was copied and modified from the online Mathworks example for
% creating an .avi animation movie from a numbered sequence of .jpg images:
%
%    http://www.mathworks.com/help/matlab/examples/convert-between-image-
sequences-and-video.html
%

workingDir = 'anim_sequences'; % set this to wherever you want
                               % You might want to turn off Dropbox or Box
                               % or iCloud or Google Drive
                               % file synchronization services while making
                               % the animation. The simulation will go faster
                               % if it doesn't also have to sync all the
                               % new image files.

% ------------------------------------
% ------  Find Image File Names  ------
% ------------------------------------
% Find all the JPEG file names in the images folder. Convert the set of image
names to a cell array.
disp(' ')
str=sprintf('using *all* images discovered in folder [%s]',workingDir);
disp(str)
disp(' ')
imageNames = dir(fullfile(workingDir,'*.jpg'));
imageNames = {imageNames.name}';
str=sprintf('discovered [%i] image files for the
animation',length(imageNames)); disp(str)

% figure out the name of the movie from the first filename
% If we've got a sequence of images named like this:
%   imageNames =
%       '2016-01-11_17_06_55_img_000001.jpg'
%       '2016-01-11_17_06_55_img_000002.jpg'
%       '2016-01-11_17_06_55_img_000003.jpg'
%       '2016-01-11_17_06_55_img_000004.jpg'
%        ... (and so on)
% Then extract all occurrences of the '_' character and make the AVI
% filename everything up to the last underbar character.
filenameStr=imageNames{1};
k = strfind(filenameStr, '_');
```

```
AviFileName_prefix=filenameStr(1:(max(k)-1));
AviFileName = strcat(AviFileName_prefix,'.avi');
str=sprintf('using animation filename: [%s]',AviFileName); disp(str)

disp(' ')
pause(4)
disp(' ')

% ------------------------------------------------------------
% ------  Create New Video with the Image Sequence  ------
% ------------------------------------------------------------
% Construct a VideoWriter object, which creates a Motion-JPEG AVI file by
default.
disp('creating a new VideoWriter() object...');
outputVideo = VideoWriter(fullfile(workingDir,AviFileName));
outputVideo.FrameRate = anim_fps;
open(outputVideo)
disp('done.');


% Loop through the image sequence, load each image, and then write it to the
video.
disp('looping through each image in the sequence')
for ii = 1:length(imageNames)
   img = imread(fullfile(workingDir,imageNames{ii}));
   writeVideo(outputVideo,img)
   str=sprintf('done processing image [%i] of [%i],
[%s]',ii,length(imageNames),imageNames{ii}); disp(str)
end
disp('done.');


% Finalize the video file.
disp('closing the outputVideo object')
close(outputVideo)
disp('done.');


disp('AVI animation file creation complete')
str=sprintf('you should now have a new .avi movie file in [%s] named
[%s]',workingDir,AviFileName); disp(str)
% you can delete the .jpg sequence files once you've got the .avi.


% don't forget to turn your file sharing services back on like Dropbox or
% Google Drive
```

[graphical_development.m]

اسکریپت کمکی m-file که برای توسعه گرافیک های عملکرد s مفید است

```
% this script is designed to be run *after* setup.m
% and prepares the workspace for a line-by-line walk through of
% sanim_XY_vehicle_viz.m
```

```
anim_frame_name_str=char(datetime('now','Format','yyyy-MM-dd_HH_mm_ss'));

% for veh_object2.m
object_type=1;
obj_length=vehicle_length;
obj_width=vehicle_width;

plotAxisLimits = plotAxisLimits;

% for mdlInitializeSizes()
Config.axisMode=plotAxisLimitMode;
Config.ax=plotAxisLimits;
Config.Ts=animation_update_interval;
Config.enable_CG_trace=enable_CG_trace;
Config.enable_rearAxle_trace=enable_rearAxle_trace;
Config.save_anim_frames=save_anim_frames;
Config.anim_frame_name_str=anim_frame_name_str;
Config.L=vehicle_length;
Config.W=vehicle_width;
%N=1;


% for mdlUpdate()

% since N=1, just put in scalars
X   = 2; % (m) object positions in global XY frame
Y   = 6; % (m) object positions in global XY frame
yaw = pi/10; % (rad) object yaw orientations about global Z axis
delta = yaw/2;

% init:
%[sys,x0,str,ts] = sanim_XY_vehicle_viz(0,0,0,0,Config)


% mdlUpdate()
%[sys,x,str,ts] = sanim_XY_vehicle_viz(0,x0,[X_ic, Y_ic, yaw_ic, 0],2,Config)
```

[sanim_XY_vehicle_viz.m]-Simulink m-file s-function

که اشیاء گرافیکی Matlab را در هر فاصله انیمیشن نمایش می دهد

(توسط anim_fps در setup.m تنظیم می شود) تا انیمیشن ایجاد شود.

```
function [sys,x0,str,ts] = sanim_XY_vehicle_viz(t,x,u,flag,Config)
% sanim_XY_vehicle_viz() - animate a 2D vehicle using SAE coordinates.
%
% This is a modified version of the Mathwork's sanim.m for animating 3D
motion.
%
% Marc Compere, comperem@gmail.com
% created : 30 July 2011
% modified: 17 Jan 2016
```

```
%
%
% Edited from the original file:
% SANIM.m S-Function for displaying 6DoF trajectories
%
% See also:  Simulink library file 'aerospace.mdl' and sanim.m
%

 % for command line development:
%x=[0 10 0 10 0]
%u=[0 0 0]


switch flag,

  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
     [sys,x0,str,ts]=mdlInitializeSizes(Config);

  %%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%
  case {1, 3, 9},
     sys=[];

  %%%%%%%%%%
  % Update %
  %%%%%%%%%%
  case 2,
     sys = [];
     sys=mdlUpdate(t,x,u,Config);

  %%%%%%%%%%%%%%%%%%%%%%%
  % GetTimeOfNextVarHit %
  %%%%%%%%%%%%%%%%%%%%%%%
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u,Config.Ts);

otherwise
  %%%%%%%%%%%%%%%%%%%%
  % Unexpected flags %
  %%%%%%%%%%%%%%%%%%%%

   error(['Unhandled flag = ',num2str(flag)]);

end

end % sanim_XY_pairs()
%
%=============================================================================
=
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
```

```
%=============================================================================
=
%
function [sys,x0,str,ts]=mdlInitializeSizes(Config)

%
% Set Sizes Matrix
%
sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 6; % x(1:4) => [xmin xmax ymin ymax] when axisMode==0
(auto),
                          % x(5)=>initState for line trace setup is ZERO at
t=0 and >0 for t>0
                          % x(6) counter for creating animation frame
sequences
sizes.NumOutputs     = 0;
sizes.NumInputs      = 4; % [X_veh, Y_veh, psi_veh, delta_steer] in the
global XY frame
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

%
% initialise the initial conditions
%
%x0  = [];

%
% str is always an empty matrix
%
str = [];

%
% initialise the array of sample times
%
%ts  = [.1 0]; % Sample period of 0.1 seconds (10Hz)
ts = [Config.Ts 0]; % inherited



%
% Initialise Figure Window
%
   h_f=findobj('type','figure','Tag','XY anim');

   if isempty(h_f)
     h_anim=figure;
   else
     h_anim=h_f;
   end

   % Figure 'position' args -> [left, bottom, width, height]
   % put the keyboard input figure right in the upper middle
```

```
   screen_size=get(0,'ScreenSize'); % [left, bottom, width, height] (in
pixels)
   set_figure_window_size=0;
   if set_figure_window_size==1,
      figure_width=672; figure_height=504; % (pixels) Matlab defaults are
560x420(?) or... 672x504 on my laptop
      figure_left = screen_size(3) - figure_width - 10; % (pixels) almost all
the way to the left side of the screen
      figure_bottom = screen_size(4) - figure_height - 100; % (pixels) almost
all the way to the top of the screen
      set(h_anim,'name','XY Animation Figure', ...
                 'renderer','OpenGL', ...
                 'clipping','off', ...
                 'position',[figure_left figure_bottom figure_width
figure_height],...
                 'Tag','XY anim');
   else
      set(h_anim,'name','XY Animation Figure', ...
                 'renderer','OpenGL', ...
                 'clipping','off', ...
                 'Tag','XY anim');
   end
%Painters
%Zbuffer
%OpenGL

   if ~isempty(h_anim)                          % if there's a figure window..
      h_del = findobj(h_anim,'type','axes'); % find the axes..
      delete(h_del);                            % delete the axes..
      figure(h_anim);                           % bring figure to the front
   end


%
% Initialize Axes
%
   handle.axes(1)=axes;
   set(handle.axes(1),...
           'visible','on','box','off', ...
           'units','normal', ...
           'position',[0.1 0.1 0.75 0.75], ...
           'Color',[.8 .8 .8], ...
           'clipping','off',...
           'XMinorTick','on',...
           'YMinorTick','on',...
           'Tag','3d_axes');

    % this reverses the direction of increasing Y values
    % to make Matlab figure window conform to SAE coordinates where:
    %    +X is to the right (as usual)
    %    +Y is down
    %    +Z is into the paper
    set(handle.axes(1),'YDir','reverse')

    % set axes to initial [xmin xmax ymin ymax]
    axis(Config.ax);
    x0=axis; % assign iniital state to current axis limits
```

```
    x0(5)=0; % initState==0 at t=0 and >0 for t>0
    x0(6)=0; % animation frame counter (only writes .jpegs if
save_anim_frames==1)
    grid on
    axis equal

    xlabel('SAE X_{gnd} (m)')
    ylabel('SAE Y_{gnd} (m)')


%
% Initialize snail trail objects (CG line trace and rearAxle trace)
%
   cmap_CG = colormap(cool(1)); % run 'colormapeditor' then choose in Tools |
Standard Colormaps for examples
   if (Config.enable_CG_trace==1),
      line_width = 2;
      handle.line_CG = line(0,0); % N times with the client using each of the
N lines for each server restart
      set(handle.line_CG,'linestyle','-
','color',cmap_CG,'userdata',0,'clipping','off','LineWidth',line_width); %
create line object for trace indicating where the Nth agent has been
   end

   cmap_rearAxle = colormap(spring(1)); % colormaps: hsv, gray, hot, cool,
copper, pink, flag, jet, autumn, spring, summer, winter
   if (Config.enable_rearAxle_trace==1),
      line_width = 2;
      handle.line_rearAxle = line(0,0); % N times with the client using each
of the N lines for each server restart
      set(handle.line_rearAxle,'linestyle','-
.','color',cmap_rearAxle,'userdata',0,'clipping','off','LineWidth',line_width
); % create line object for trace indicating where the Nth agent has been
   end

%
% Initialize vehicle object trajectories (chassis, front and rear tires)
%
   % make a generic set of vertices and faces for a 2D vehicle object
   veh=veh_object2(1,Config.L,Config.W); % see 'help patch' for how to make
patch graphics objects
   cmap_veh = colormap(summer(1)); % colormaps: hsv, gray, hot, cool, copper,
pink, flag, jet, autumn, spring, summer, winter
   handle.veh =
patch('Vertices',veh.vertices','Faces',veh.faces,'AmbientStrength',0.46,'Face
Color',cmap_veh,'EdgeColor',[0 0 0],'FaceAlpha',0.1);
   X_loc=sum(veh.vertices(1,:))/length(veh.vertices(1,:));
   Y_loc=sum(veh.vertices(2,:))/length(veh.vertices(2,:));
   handle.veh_text =
text(X_loc,Y_loc,'veh','FontSize',10,'HorizontalAlignment','center','Vertical
Alignment','middle'); % default FontSize is 10

   % create tire vertices, then make 4 different graphics patch objects to
move around independently
   tire=veh_object2(2,Config.L/5,Config.W/5);
```

```
   handle.tire_rf =
patch('Vertices',tire.vertices','Faces',tire.faces,'AmbientStrength',0.46,'Fa
ceColor',cmap_veh,'EdgeColor',[1 1 1],'FaceAlpha',0.1);
   handle.tire_lf =
patch('Vertices',tire.vertices','Faces',tire.faces,'AmbientStrength',0.46,'Fa
ceColor',cmap_veh,'EdgeColor',[1 1 1],'FaceAlpha',0.1);
   handle.tire_rr =
patch('Vertices',tire.vertices','Faces',tire.faces,'AmbientStrength',0.46,'Fa
ceColor',cmap_veh,'EdgeColor',[1 1 1],'FaceAlpha',0.1);
   handle.tire_lr =
patch('Vertices',tire.vertices','Faces',tire.faces,'AmbientStrength',0.46,'Fa
ceColor',cmap_veh,'EdgeColor',[1 1 1],'FaceAlpha',0.1);


   % we must store all unmodified vertices in their local coordinate
   % frame so mdlUpdate() can orient and position in the global
   % then store them in the AXES UserData (not the figure window's UsersData)
   pass_these_verts{1} = get(handle.veh ,'Vertices');
   pass_these_verts{2} = get(handle.tire_rf,'Vertices');
   set(handle.axes(1),'userdata',pass_these_verts); % store veh object
vertices for retrieval in mdlUpdate() below

%
% Set Handles of graphics in FIGURE UserData
%
   set(h_anim,'UserData',handle); % store axes, line, and veh objects just
created for retrieval in mdlUpdate()

end % mdlInitializeSizes
%
%=============================================================================
=
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=============================================================================
=
%
function sys=mdlUpdate(t,x,u,Config);

sys=x; % initialize outputs

X     = u(1); % (m) object positions in global XY frame
Y     = u(2); % (m) object positions in global XY frame
yaw   = u(3); % (rad) vehicle's yaw orientations about global Z axis
delta = u(4); % (rad) front tire's steered angle (w.r.t. vehicle yaw angle)

%
% Retrieve figure object handles
%
    handle = get(findobj('type','figure','Tag','XY anim'),'userdata'); %
retrieve all graphics objects handles

    if isempty(findobj('type','figure','Tag','XY anim'))
     %figure has been manually closed
```

```
     return
    end

%
% Update all object positions
%

   if ~isempty(handle.axes(1))
      % retrieve the object vertices from the figure window's AXES UserData
      pass_these_verts=get(handle.axes(1),'UserData');
      if isempty(pass_these_verts)
         %axes userdata is missing for some reason - exit
         return
      end
   else
      % no axes handle for some reason - exit
      return
   end

   % pull out the veh vertices
   veh_verts    = pass_these_verts{1};
   tire_verts   = pass_these_verts{2};

   % now do all the same translation and rotation for the vehicle objects
   % ----------------------------------------------------
   % first: retrieve the i'th object's vertices (vehicle and rear two tires
all have same)
   verts_veh_xy     = [veh_verts]; % pick off the i'th [X,Y] column pair
   verts_tire_rf_xy = [tire_verts]; % all 4 tires use identical vertices
prior to XY positioning
   verts_tire_lf_xy = [tire_verts];
   verts_tire_rr_xy = [tire_verts];
   verts_tire_lr_xy = [tire_verts];
   [a_veh,b]  = size(verts_veh_xy);     % a_veh is number of vertices in
vehicle object (10 for vehicle_object2(1,[]) )
   [a_tire,b] = size(verts_tire_rf_xy); % b_vehicle is number of vertices in
tire object (12 for vehicle_object2(2,[]) )

   attitude         = [ cos(yaw)   -sin(yaw)   ;  sin(yaw)    cos(yaw)  ]; %
transformation matrix from body-fixed to global or terrain frame
   attitude_steered = [ cos(delta) -sin(delta) ;  sin(delta)  cos(delta)]; %
transformation matrix from body-fixed to global or terrain frame

   % do the schmack: translate in body-fixed xy, rotate about yaw(i) with
'attitude', then translate in XY to the terrain frame's [X(i),Y(i)] position
   verts_veh_XY     = attitude*[verts_veh_xy' ]
+ [ X ; Y]*ones(1,a_veh );
   verts_tire_rr_XY = attitude*[verts_tire_rr_xy' + [ -Config.L/2 ; -
Config.W/2]*ones(1,a_tire)] + [ X ; Y]*ones(1,a_tire);
   verts_tire_lr_XY = attitude*[verts_tire_rr_xy' + [ -Config.L/2 ;
+Config.W/2]*ones(1,a_tire)] + [ X ; Y]*ones(1,a_tire);

   % front tires require special consideration: rotate by steer angle first,
then translate in xy, rotate by yaw, then translate in XY
   verts_tire_rf_xy_steered = attitude_steered*[verts_tire_rf_xy'];
```

```
   verts_tire_rf_XY = attitude*[verts_tire_rf_xy_steered + [Config.L/2;-
Config.W/2]*ones(1,a_tire)] + [X;Y]*ones(1,a_tire);

   verts_tire_lf_xy_steered = attitude_steered*[verts_tire_lf_xy'];
   verts_tire_lf_XY = attitude*[verts_tire_lf_xy_steered +
[Config.L/2;+Config.W/2]*ones(1,a_tire)] + [X;Y]*ones(1,a_tire);


   % update the figure window object with the new position and orientation
   set(handle.veh,'Vertices',verts_veh_XY');
   set(handle.veh_text,'Position',[X;Y]);

   set(handle.tire_rf,'Vertices',verts_tire_rf_XY'); % set graphics handle
vertices to vertices just rotated and translated
   set(handle.tire_lf,'Vertices',verts_tire_lf_XY');
   set(handle.tire_rr,'Vertices',verts_tire_rr_XY');
   set(handle.tire_lr,'Vertices',verts_tire_lr_XY');




%
% Update Line Objects
%
   if (Config.enable_CG_trace==1),
      initState = x(5); % 0 the first time through only
      %str=sprintf('sanim_tracked_vehicle.m:
initState=%i',initState);disp(str)
      if initState>=1, % tack on the current vehicle positions to the vehicle
line trace
         xLine = get(handle.line_CG,'XData');
         yLine = get(handle.line_CG,'YData');

         % use the graphics line objects XData and YData to store and display
a growing set of line points
         set(handle.line_CG,'Xdata',[xLine X],'Ydata',[yLine Y]);

      else % init==0 so create first line point from vehicle IC's coming in
from the UDP client s-function (not the x0 initialized with zeros in this s-
function)

         sys(5)=1; % cause 'initState' to no longer be zero

         set(handle.line_CG,'Xdata',X,'Ydata',Y);

      end
   end % if Config.enable_CG_trace==1


   if (Config.enable_rearAxle_trace==1),
      initState = x(5); % 0 the first time through only
      %str=sprintf('sanim_tracked_vehicle.m:
initState=%i',initState);disp(str)
```

```
      if initState>=1, % tack on the current vehicle positions to the vehicle
line trace

            line_rearAxle_verts = attitude*[-Config.L/2 ; 0 ] + [X;Y]; %
[X,Y] pair in i'th column

            xLine = get(handle.line_rearAxle,'XData');
            yLine = get(handle.line_rearAxle,'YData');

            % use the graphics line objects XData and YData to store and
display a growing set of line points
            set(handle.line_rearAxle,'Xdata',[xLine
line_rearAxle_verts(1)],'Ydata',[yLine line_rearAxle_verts(2)]);

      else % init==0 so create first line point from vehicle IC's coming in
from the UDP client s-function (not the x0 initialized with zeros in this s-
function)

         sys(5)=1; % cause 'initState' to no longer be zero

         line_rearAxle_verts = attitude*[-Config.L/2 ; 0 ] + [X;Y]; % [X,Y]
pair in i'th column


set(handle.line_rearAxle,'Xdata',line_rearAxle_verts(1),'Ydata',line_rearAxle
_verts(2));

      end
   end % if Config.enable_rearAxle_trace==1


   % ------------------------------------------
   % --------  update the axis limits  --------
   % ------------------------------------------
   sys(1:4)=x(1:4); % init the first 4 discrete state (updated immmediately
below)

   % this is where we grow the axis limits but never shrink - it captures
   % all objects and zooms out but does not pan or shrink limits (looks
better)
   if (Config.axisMode==0), % -> GROW-TO-FIT from initial user-supplied axis
limits
      axis tight % this resets the axes to capture all objects
      axisTight=axis; % capture those new minimum limits
      sys(1)=min(x(1),axisTight(1)); % new xmin as smaller of auto or user-
specified
      sys(2)=max(x(2),axisTight(2)); % new xmax as larger of auto or user-
specified
      sys(3)=min(x(3),axisTight(3)); % new ymin as smaller of auto or user-
specified
      sys(4)=max(x(4),axisTight(4)); % new ymax as larger of auto or user-
specified

      % at this point the axis limits are captured but not assigned..
```

```
      axis(sys(1:4)); % so make the assignment to the graphics window

   else, % -> FIXED-ONLY from user supplied initial axis limits
      axis(Config.ax);
   end

%
% Force MATLAB to Update Drawing
%
   %set(handle.axes(1),'visible','off')
   %drawnow

% make a sequence of animation frames for a movie?
if Config.save_anim_frames==1,

   % increment state x(6) for current frame number
   sys(6) = x(6) + 1;

   % create jpg filename string using datetime() function in Simulink
   % dialogue box - this creates a single animation sequence using the date
   % and time from when the Simulink model was started
   imgFileStr=sprintf('%s_img_%0.6i.jpg',Config.anim_frame_name_str,sys(6));
% note: %0.6i pads with leading zeros just like writeVideo() demo

   % prepend a folder to contain all these animation sequence images
   myFile = fullfile('anim_sequences',imgFileStr);

   str=sprintf('saving image sequence [%s]',myFile); disp(str)

   % write this graphics frame to a file
   print('-opengl','-djpeg',myFile);
end

end % mdlUpdate

%
%=============================================================================
=
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block.
%=============================================================================
=
%
%function sys=mdlGetTimeOfNextVarHit(t,x,u,Ts)

%    sys = ceil(t/Ts)*Ts+Ts;

% end mdlGetTimeOfNextVarHit
```

[veh_object2.m]

پشتیبانی از عملکرد فایل m برای ایجاد رأس و چهره برای وسایل نقلیه و تایرهای گرافیکی

```
function out=veh_object2(object_type,obj_length, obj_width)
%
```

```
% This file was developed from a copy of wheel_object.m.
%
% Generate vertices and faces for a top-view of a simple 2D vehicle patch
% object or tire.
%
% Assumed CG is at object's geometric center.


L = obj_length; % (m) L for vehicle or tire (depending on object_type)
W = obj_width; % (m) width of vehicle or tire


% -------------------------------------------------------------
% -------------------- vehicle object -------------------------
% -------------------------------------------------------------
% define 2D top-down view of vehicle body, starting from the nose, centered
% at vehicle's geometric center, g
% pt# 1       2     3           4      5     6        7       8     9  10
x_veh = (1/2)*[ 1,  0.95,   0.85,      -0.9,    -1,   -1,    -0.9,   0.85,
0.95,  1];
y_veh = (1/2)*[ 0,   0.6,      1,         1,   0.8, -0.8,      -1,     -1, -
0.6,  0];

%x_veh = x_veh + 0.5; % (m) translate all vertices forward such that the
%                     %     origin (x,y)=(0,0) is at the rear axle center


% make the vehicel chassis a little longer than the wheelbase
veh_obj.vertices = 1.3*[L*x_veh;
                        W*y_veh];

% define the object faces
veh_obj.faces = 1:length(x_veh);



% -------------------------------------------------------------
% --------------------  tire object -------------------------
% -------------------------------------------------------------
% define 2D top-down view of vehicle body, starting from the nose, centered
% at geometric center of tire
% pt#        1       2     3         4      5     6         7      8     9
10     11     12
x_tire = (1/2)*[ 1   ,  0.98,  0.95,    -0.95, -0.98,   -1,      -1, -0.98, -
0.95,    0.95,  0.98,     1];
y_tire = (1/2)*[ 0.60,  0.90,     1,        1,  0.90, 0.60,    -0.60, -0.90,
-1,      -1, -0.90, -0.60];

L_tire = obj_length;
W_tire = obj_width;
tire_obj.vertices = [L_tire*x_tire;
                     W_tire*y_tire];

% define the object faces
```

```
tire_obj.faces = 1:length(x_tire);
```

```
% Assign the function output
if object_type==1, % output vehicle vertices and faces
   out.vertices = veh_obj.vertices;
   out.faces    = veh_obj.faces;
elseif object_type==2, % output tire vertices and face
   out.vertices = [ tire_obj.vertices ];
   out.faces    = [ tire_obj.faces ];
end
```

```
% draw?
draw=0;
if draw==1,
   clf
   veh_handle =
patch('Vertices',out.vertices','Faces',out.faces,'FaceColor',[0.5 0.5
0.5],'EdgeColor',[0 0 1],'FaceAlpha',0.1);
   xlabel('X-axis')
   ylabel('Y-axis')
   zlabel('Z-axis')
   axis equal
   grid on
end
```

Reference

[1] A. I. Aulia, M. Faswia Fahmi, H. Hindersah, A. S. Rohman and E. Muhammad Idris Hidayat, "Implementation of Motion Cueing and Motor Position Control for Vehicle Simulator with 4-DOF-Platform," *2019 6th International Conference on Electric Vehicular Technology (ICEVT)*, 2019, pp. 40-45, doi: 10.1109/ICEVT48285.2019.8994028.

[2] I. Tanev, M. Joachimczak, H. Hemmi and K. Shimohara, "Evolution of the driving styles of anticipatory agent remotely operating a scaled model of racing car," 2005 IEEE Congress on Evolutionary Computation, 2005, pp. 1891-1898 Vol. 2, doi: 10.1109/CEC.2005.1554918.

[3] S. Jin, J. Kim, J. Kim and T. Seo, "Six-Degree-of-Freedom Hovering Control of an Underwater Robotic Platform With Four Tilting Thrusters via Selective Switching Control," in IEEE/ASME Transactions on Mechatronics, vol. 20, no. 5, pp. 2370-2378, Oct. 2015, doi: 10.1109/TMECH.2014.2378286.

[4] Y. Zhang and Y. Yao, "Optimal Design of 6-DOF Parallel Robot Based on Output Frequency Response Function," 2009 International Conference on Measuring Technology and Mechatronics Automation, 2009, pp. 841-844, doi: 10.1109/ICMTMA.2009.624.

[5] M. bin Mansor, K. Hudha, Z. Kadir and N. H. Amer, "Active front wheel steering system for 14 DOF armoured vehicle model due to firing force disturbance," 2015 10th Asian Control Conference (ASCC), 2015, pp. 1-6, doi: 10.1109/ASCC.2015.7244493.

[6] H. M. Do, C. Park, B. I. Kim and G. J. Chung, "Development of simulation model for 6 DOF parallel robot," 2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2013, pp. 333-334, doi: 10.1109/URAI.2013.6677380.

[7] E. Wilhelm, L. Rodgers and R. Bornatico, "Real-time electric vehicle mass identification," 2013 World Electric Vehicle Symposium and Exhibition (EVS27), 2013, pp. 1-6, doi: 10.1109/EVS.2013.6914840.

[8] T. Du, K. Wang and Y. Li, "A tracking system based on 2-DOF motion platform for flight simulator," 2020 39th Chinese Control Conference (CCC), 2020, pp. 2723-2728, doi: 10.23919/CCC50068.2020.9189479.

[9] D. Ning, H. Du, S. Sun, W. Li and B. Zhang, "An Innovative Two-Layer Multiple-DOF Seat Suspension for Vehicle Whole Body Vibration Control," in IEEE/ASME Transactions on Mechatronics, vol. 23, no. 4, pp. 1787-1799, Aug. 2018, doi: 10.1109/TMECH.2018.2837155.

[10] Y. Xu et al., "Research on Motion Algorithm for Intelligent Operation of Rock Drill With Eight Degrees of Freedom," 2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR), 2018, pp. 61-65, doi: 10.1109/IISR.2018.8535962.