شبیه‌سازی و پیاده‌سازی وظایف دوری از موانع و همگام‌سازی با استفاده از تئوری کنترل سوپروایزری احتمالی و زمان‌دار در رباتیک جمعی

فائزه میرزائی[۱،*]، علی‌اکبر پویان[۲] و سعیده فردوسی[۳]

اطلاعات مقاله

دریافت مقاله: ۱۳۹۶/۱۱/۰۳

پذیرش مقاله: ۱۳۹۶/۱۲/۲۶

واژگان کلیدی:

رباتیک جمعی،

تئوری کنترل سوپروایزری احتمالی و زمان‌دار،

وظیفه دوری از موانع،

وظیفه همگام‌سازی،

سکوی آرگوس.

چکیده

به‌دلیل دشواری پیاده‌سازی صوری سیستم‌های مبتنی بر رباتیک جمعی، نرم‌افزارهای کنترلی این سیستم‌ها اغلب با آزمون و خطا و بصورت تک‌منظوره طراحی می‌شوند. بنابراین، طراحی صورت‌گرفته به‌سادگی قابلیت استفاده مجدد برای سایر مسائل ندارد. از طرفی، آزمون، آنالیز و تصدیق وجود ویژگی‌های موردنیاز در مسئله به‌دشواری صورت می‌گیرد. علاوه براین، به‌دلیل تولید دستی کد، نمی‌توان تطبیق طراحی اولیه و کد پیاده‌سازی‌شده را تضمین کرد. برای غلبه بر این چالش‌ها، استفاده از تئوری کنترل سوپروایزری پیشنهاد می‌شود. در این مقاله، تئوری کنترل سوپروایزری احتمالی و زمان‌دار ptSCT بر روی سکوی نوظهور آرگوس (ARGoS) و در بستر رباتیک جمعی پیاده‌سازی شده است. روش پیشنهادی، پس از محاسبه سوپروایزر احتمالی و زمان‌دار، کد کنترلی موردنیاز را به‌صورت خودکار تولید می‌کند. این کد بدون هیچ تغییری قابل‌استفاده برای شبیه‌سازی یا اجرا بر روی ربات‌های واقعی است. برای نمایش قابلیت‌های روش پیشنهادی، دو وظیفه پرکاربرد دوری از موانع (Obstacle Avoidance) و همگام‌سازی (Synchronization) با تکیه بر طراحی احتمالی و زمان‌دار، پیاده‌سازی و شبیه‌سازی شده‌اند. سپس مزایای طراحی با ptSCT نسبت به SCT مورد اشاره قرار گرفته است. نتایج آزمایش‌های صورت‌گرفته نشان از عملکرد بسیار مناسب روش پیشنهادی از نظر سادگی و صحت طراحی، قابلیت استفاده مجدد و تولید خودکار کد دارد.

۱- مقدّمه

رباتیک جمعی[۴] (SR) از موضوعاتی است که در دهه اخیر، موردتوجه محققان بسیاری قرار گرفته است [۱–۵]. رباتیک جمعی، حوزه‌های هوش مصنوعی، تئوری کنترل، رباتیک، مهندسی سیستم و بیولوژی را در کنار یکدیگر قرار می‌دهد. در رباتیک جمعی به مطالعهٔ جمع بزرگی از ربات‌ها پرداخته می‌شود که برای انجام یک هدف مشترک با هم همکاری می‌کنند. برخی از محققان علم رباتیک، از هوش انسان و برخی دیگر از رفتار حیوانات در طبیعت الهام می‌گیرند[۳]. رباتیک جمعی در تلاش است تا طرز فکر و رفتار حیوانات را در موقعیت‌های مختلف تقلید کند. رباتیک جمعی دارای ویژگی‌های خاصی است که آن را از سیستم‌های چندرباته[۵]متمایز می‌کند. ویژگی‌هایی همچون استقلال[۶]، تعداد زیاد[۷]، توانایی محدود، مقیاس‌پذیری[۸]، مقاومت[۹]،

* پست الکترونیک نویسنده مسئول: fmirzaei@shahroodut.ac.ir

۱. دانشجوی دکتری، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی شاهرود

۲. دانشیار، دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی شاهرود

۳. استادیار، دانشکده برق و رباتیک، دانشگاه صنعتی شاهرود

4.Swarm Robotic
5.Multi-robot Systems
6.Autonomy
7.Large Number
8.Scalability
9.Robustness

انعطاف[1]، هماهنگی توزیع‌شده[2] و همگن بودن[3] از مهم‌ترین این موارد هستند [۴]، درصورتی‌که در سیستم‌های تک‌رباته و چندرباته چنین مواردی یا وجود ندارند یا به قوّت رباتیک جمعی دیده نمی‌شوند [۶–۸]. در رباتیک جمعی، ربات‌ها باید بتوانند به‌صورت فیزیکی با محیط تعامل داشته، بر آن اثر بگذارند. در این سیستم‌ها، مشابه اجتماعات حیواناتی مانند مورچه و زنبور عسل، شاهد تجمع تعداد زیادی ربات هستیم.

علاوه براین، رهبر کنترل‌کننده‌ای نیز مشاهده نمی‌شود. قابلیت‌های هریک از ربات‌ها باید کم باشد، به گونه‌ای که توانایی انجام وظایف بزرگ را به‌تنهایی نداشته باشند. این مسئله به کاهش هزینهٔ ربات نیز کمک می‌کند که از قیود اصلی رباتیک جمعی است. سیستم باید با افزایش تعداد ربات‌ها به‌درستی عمل کند و حتی در شرایط مناسب، کارایی کل مجموعه ارتقا یابد.

ربات‌ها توانایی تولید راه‌حل‌های پیمانه‌ای[4] برای وظایف مختلف دارند؛ به این معنی که برای انجام عملیات مختلف، نیاز به تنظیمات سخت‌افزاری زیاد یا تغییر سکو[5]های نرم‌افزاری نباشد. مجموعه ربات‌ها در شرایطی که تعدادی از ربات‌ها دچار خسارت شده یا محیط آشوبی شود، باید به کار خود ادامه دهد، حتی اگر لازم باشد کارایی تا حدی کاهش یابد. درواقع، با وجود فراوانی[6] در این سیستم‌ها، خسارت یک ربات، توسط ربات‌های دیگر جبران می‌شود. در رباتیک جمعی، هماهنگی بین ربات‌ها توزیع شده است و هر ربات حسگرها و ارتباطات محدود و محلی دارد. همچنین، تعداد محدودی گروه وجود دارد که ربات‌های هر گروه از نظر فیزیکی و نرم‌افزاری مشابه‌اند و نقش یکسان در عملیات دارند. طراحی منطق کنترل‌کننده ربات‌ها (سوپروایزرها) یک مسئله چالش‌برانگیز است؛ زیرا منطق در سطح ربات‌ها و به‌صورت منفرد طراحی می‌شود، ولی نتیجهٔ آن به شکل جمعی موردبررسی قرار می‌گیرد. از طرفی، همه ربات‌ها یک کد کنترلی یکسان را اجرا می‌کنند. به همین دلیل، نرم‌افزار کنترل‌کننده ربات‌ها معمولاً به روش‌های خاص منظوره[7] و اغلب با استفاده از روش‌های مبتنی بر مهندسی نرم‌افزار حاصل می‌شود. سیستم‌های طراحی‌شده به این روش‌ها به‌سادگی قابل انتقال به کاربردهای دنیای واقعی نیستند. علاوه براین، تجزیه‌وتحلیل وجود ویژگی‌های موردنیاز کنترل‌کننده در کدهای به‌دست‌آمده از این روش‌ها دشوار بوده، نگهداری از این کدها نیز به‌سادگی صورت نمی‌پذیرد [۹].

حتی در صورت استفاده از روش‌های صوری[8] برای طراحی منطق کنترل‌کننده، هیچ تضمینی برای انطباق کدهای نوشته‌شده با طراحی صورت‌گرفته وجود ندارد [۱۰].

مرجع [۱۱] روشی با عنوان طراحی ویژگی‌محور[9] ارائه کرده است. این روش، شامل چهار مرحله است. ابتدا نیازها به‌صورت صوری تعیین می‌شوند. در بخش دوم، مدل ماکروسکوپی[10] با استفاده از زنجیره‌ی مارکو[11] طراحی و توسط روش بررسی مدل[12] ارزیابی شده، سپس از مدل برای پیاده‌سازی و شبیه‌سازی استفاده می‌شود. مرجع [۱۲] روشی برای طراحی خودکار ماشین حالت متناهی احتمالاتی ارائه داده است. در این روش، از پیمانه‌های ازپیش‌تعریف‌شده‌ای برای طراحی استفاده می‌شود، به‌طوری‌که حالت‌های مناسب طراحی ماشین حالت متناهی توسط روش‌های بهینه‌سازی تعیین می‌شوند.

شبکه پتری[13] روشی دیگر برای طراحی کنترل‌کننده در رباتیک جمعی است [۱۳]. کنترل‌کننده‌های مبتنی بر شبکه پتری توسط یک کامپیوتر مرکزی با ربات‌ها در ارتباط هستند، درصورتی‌که در رباتیک جمعی نباید کنترل‌کنندهٔ مرکزی وجود داشته باشد؛ زیرا این امر موجب افزایش هزینه و پیچیدگی طراحی می‌شود. منطق زمانی[14] نیز برای مدل کردن و آنالیز سیستم‌های رباتیک جمعی استفاده شده است [۱۴]. این روش می‌تواند با روش بررسی مدل ترکیب شده، ارزیابی صوری سیستم را انجام دهد[۱۴].

استفاده از تئوری کنترل سوپروایزری[15] (SCT) در رباتیک جمعی اولین بار در سال ۲۰۱۶ مطرح شد [۹]. چند مورد از مزایای SCT در مقابل استفاده از سایر روش‌های صوری موجود عبارت‌اند از: کاهش راه‌حل‌های خاص منظوره، تولید خودکار کدهای کنترلی برای مدل‌سازی نیازهای[16] سیستم،

[1].Flexibility
[2].Distributed Coordination
[3].Homogenous
[4].Modular
[5].Platform
[6].Redundancy
[7].Ad Hoc
[8].Formal

[9].Property-driven Design
[10].Macroscopic
[11].Markov Chain
[12].Model Checking
[13].Petri Net
[14].Temporal Logic
[15].Supervisory Control Theory
[16].Specification

شکستن طراحی یک سیستم پیچیده به بخش‌های کوچک و قابل‌فهم، اثبات وجود رفتارها[1] در کد کنترلی و قابلیت استفادهٔ مجدد[2] سوپروایزرهای طراحی‌شده در سکوهای رباتیک مختلف. در بخش بعد به این موارد بیشتر پرداخته خواهد شد. نسخه احتمالی از تئوری کنترل سوپروایزری برای اولین بار توسط لوپز و همکاران ارائه گردید که در این مقاله به توسعه آن پرداخته شده است[۱۵].

روش پیشنهادی در این مقاله دارای چند نوآوری است. در گام اول، تئوری کنترل سوپروایزری احتمالی و زمان‌دار ptSCT در رباتیک جمعی معرفی شده است. آن‌گاه، سیستمی برای طراحی رفتارها و مشخصه‌های کنترلی سیستم به روش صوری پیاده‌سازی شده است. سیستم پیشنهادی به‌صورت خودکار و با استفاده از رفتارها و مشخصه‌های کنترلی سیستم، کنترل‌کننده را محاسبه می‌کند. آن‌گاه وجود ویژگی‌هایی مانند کنترل‌پذیری و عدم وجود بن‌بست را به روش‌های صوری بررسی و تضمین می‌کند. پشتیبانی از رخدادهای احتمالی و زمان‌دار، از ویژگی‌های بارز سیستم پیشنهادی است. در گام سوم، کد کنترلی معادل با سوپروایزر به‌دست‌آمده در مرحله قبل به‌صورت خودکار تولید می‌شود. در گام چهارم، تئوری کنترل سوپروایزری احتمالی و زمان‌دار ptSCT در سکوی نرم‌افزاری آرگوس[3] [۱۶] پیاده‌سازی می‌شود. به این وسیله، کد به‌دست‌آمده در مرحله قبل، به‌صورت مستقیم در سکوی آرگوس در حالت شبیه‌سازی یا اجرای واقعی قابل‌استفاده خواهد بود. در پایان، دو روش جدید به‌ترتیب برای پیاده‌سازی وظیفه‌های دوری از موانع[4] و همگام‌سازی[5] ربات‌ها با استفاده از ptSCT پیشنهاد شده که بر پشتیبانی سیستم از رخدادهای احتمالی و زمان‌دار تأکید دارند.

در ادامهٔ مقاله و در بخش دوم، به معرفی تئوری کنترل سوپروایزری پرداخته شده است. پیاده‌سازی تئوری کنترل سوپروایزری احتمالی و زمان‌دار در رباتیک جمعی در بخش سوم تشریح شده است. بخش چهارم و پنجم به بیان جزئیات پیاده‌سازی دو وظیفه مورد اشاره با استفاده از ptSCT اختصاص دارد. بخش ششم شامل نتایج شبیه‌سازی‌های صورت‌گرفته است. نتیجه‌گیری نیز در بخش پایانی ارائه شده است.

۲- تئوری کنترل سوپروایزری (SCT)

تئوری کنترل سوپروایزری یک چارچوب تئوری برای ترکیب[6] کنترل‌کننده‌ها (سوپروایزرها) است. فرض می‌شود سیستم موردبررسی را می‌توان به‌صورت یک سیستم رخداد گسسته[7] (DES) بیان کرد. یک سیستم رخداد گسسته، از تعدادی حالت گسسته تشکیل شده، جابه‌جایی بین حالات توسط رخدادها صورت می‌گیرد که گذار[8] نام دارد. SCT بین رخدادهای قابل‌کنترل و رخدادهای غیرقابل‌کنترل، تمایز قائل می‌شود. رخدادهای غیرقابل‌کنترل درواقع سیگنال‌های واکنشی همچون سیگنال ارسال‌شده توسط حسگرها هستند. رخدادهای قابل‌کنترل، سیگنال‌های فرمان هستند که توسط کنترل‌کننده ایجاد می‌شوند.

در SCT، طراح دو مسئله را مدل می‌کند: ۱. رفتارها: سیستم چه کاری می‌تواند انجام دهد؟ ۲. نیازها: سیستم چه کاری را باید انجام دهد؟ در بخش اول، توانایی‌های سیستم توسط تعدادی مدل رفتار آزاد[9] توصیف و در بخش دوم مشخصه‌های کنترلی[10] تعیین می‌شوند. هر دو بخش مذکور توسط یک زبان صوری تعریف می‌شود. زبان مجموعه‌ای از کلمات است و کلمات نتیجهٔ الحاق الفبای زبان. هر حرف الفبا به یک رویداد در DES نسبت داده می‌شود؛ بنابراین، دنبالهٔ مطلوب رویدادها، کلمات زبان را تشکیل می‌دهند. SCT همهٔ مدل‌های رفتار آزاد (رفتارها) و مشخصات کنترلی (نیازها) را در یک زبان ترکیب می‌کند. ناظر SCT با محدودکردن مجموعه‌ای از رخدادهای قابل‌کنترل که سیستم انتخاب می‌کند، تضمین می‌نماید که در هر زمان، تنها کلمات معتبر یا پیشوند کلمات معتبر اتفاق بیفتد.

فرض کنید رباتی می‌خواهد یک شیشه شیر را از یخچال بردارد. ربات در ابتدا رویداد قابل‌کنترل «باز کردن در یخچال» را انتخاب می‌کند. حال اگر رویداد غیرقابل‌کنترل «شیر در یخچال وجود دارد»، اتفاق بیفتد؛ SCT مجموعه رویدادها را به «شیر را از یخچال بردار» و «در یخچال را ببند» محدود می‌کند. درصورتی‌که رویداد غیرقابل‌کنترل «شیر در یخچال وجود ندارد» اتفاق بیفتد، SCT مجموعه

[1].Properties
[2].Reusability
[3].ARGoS
[4].Obstacle Avoidance
[5].Synchronization
[6].Synthesis
[7].Discrete Event System
[8].Transition
[9].Free Behavior Model
[10].Control Specification

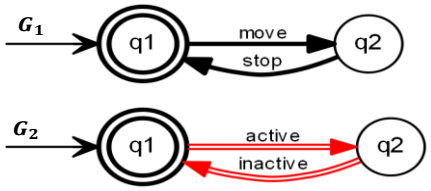
رویدادها را به «در یخچال را ببند» محدود می‌کند. در هر دو سناریو، ربات تا بسته‌شدن در یخچال، کار دیگری نمی‌تواند انجام دهد.

۲-۱- مولّدها

زبان‌های منظم[1] رایج‌ترین زبان‌های استفاده‌شده در SCT هستند که تحت عنوان زبان‌های نوع سوم نیز شناخته می‌شوند. کلمات زبان منظم را می‌توان توسط مولّد[2] ایجاد کرد. مولّد ساختاری شبیه به آتاماتای متناهی داشته، ماشین حالت متناهی[3] خوانده می‌شود. آتاماتا کلمات یک زبان را با «پذیرش» یا «رد» آن‌ها تشخیص می‌دهد، ولی مولّد کلمات یک زبان را تولید می‌کند. یک مولّد G به‌صورت $(Q,\Sigma,\delta,q_0,Q_m)$ تعریف می‌شود که در آن Q مجموعهٔ حالات، Σ الفبای زبان و δ تابع انتقال است که به‌صورت $\delta: Q\times\Sigma\to Q$ تعریف می‌شود. $q_0\in Q$ حالت اولیه و $Q_m\subseteq Q$ مجموعه حالات علامت‌دار هستند. حالات علامت‌دار به حالت‌های امن سیستم گفته می‌شوند. به‌عنوان مثال، پایان یک وظیفه را می‌توان با حالت علامت‌دار نشان داد، ولی این حالت‌ها لزوماً به‌معنی پایان عملیات نیست و ممکن است سیستم پس از این حالت‌ها، به کار خود ادامه دهد. زبان تولیدشده توسط مولّد G را به‌صورت $L(G)$ نمایش می‌دهند. رخدادهای سیستم که معادل الفبای زبان هستند، به دو نوع قابل‌کنترل (Σ_c) و غیرقابل‌کنترل (Σ_u) تقسیم می‌شوند، به‌صورتی‌که $\Sigma=\Sigma_c\cup\Sigma_u$ و $\Sigma_c\cap\Sigma_u=\emptyset$. رویداد قابل‌کنترل $e_c\in\Sigma_c$ در حالت $q\in Q$ تنها در صورتی فعال می‌شود که تابع $\delta(q,e_c)$ تعریف شده باشد.

۲-۲- مدل رفتار آزاد

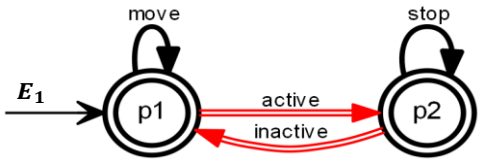
در تئوری کنترل سوپروایزری، هریک از توانایی‌های فیزیکی سیستم (رفتارها) با استفاده از یک مدل رفتار آزاد بیان می‌شود. در نتیجه، برای یک سیستم، m مولّد G_i $(i\in\{1,2,..m\})$ تعریف شده که هر مولّد یک مدل رفتار آزاد را توصیف می‌کند. مدل‌های رفتار آزاد را به‌صورت پیش‌فرض، مستقل از یکدیگر در نظر می‌گیرند. نوار نقاله‌ای را در یک کارخانهٔ تولیدی در نظر بگیرید که محصولات را جابه‌جا می‌کند. در کنار نوار نقاله، حسگری وجود دارد که وجود یا عدم وجود محصول روی نوار نقاله را تشخیص می‌دهد. شکل (۱) مدل رفتار آزاد مربوط به نوار نقاله و حسگر را نشان می‌دهد. رخدادهای *move* و *stop* قابل‌کنترل و رخدادهای *active* و *inactive* غیرقابل‌کنترل هستند. معمولاً تنها حالت اولیهٔ مدل‌های رفتار آزاد را علامت‌دار قرار می‌دهند. این کار به‌معنی برگشت سوپروایزر به شرایط اولیه است. کمان‌های یک‌خطی و دوخطی به‌ترتیب به‌معنی رویدادهای قابل‌کنترل و غیرقابل‌کنترل هستند.



شکل ۱: مثالی از مدل رفتار آزاد نوار نقاله (G1) و حسگر (G2) [۱۷]

۲-۳- مشخّصه‌های کنترلی

رفتار مطلوب سیستم با n مشخصه کنترلی نمایش داده می‌شود. هر مشخصهٔ کنترلی E_j $(j\in\{1,2,..n\})$ ارتباط دو یا چند مدل رفتار آزاد را نشان می‌دهد. شکل ۲ مشخصهٔ تعیین‌کنندهٔ ارتباط بین مدل‌های نوار نقاله و حسگر را نمایش می‌دهد. قانون موجود در این مشخصه به‌صورت «هنگامی که یک محصول در مقابل حسگر قرار گیرد، نوار نقاله متوقف شد، درغیراین‌صورت باید حرکت کند» است. SCT از وقوع برخی رخدادهای قابل‌کنترل در بعضی از حالت‌ها جلوگیری می‌کند. در مثال نوار نقاله و حسگر، در حالت p_1 رخداد *stop* غیرفعال و رخداد *move* فعال است؛ بنابراین هنگامی که حسگر غیرفعال است، نوار حرکت می‌کند.



شکل ۲: مشخصه کنترلی که موجب می‌شود نوار نقاله تنها در صورتی حرکت کند که محصولی در مقابل حسگر نباشد [۱۷]

۳- تئوری کنترل سوپروایزری ptSCT در رباتیک جمعی

روش پیشنهادی، تئوری کنترل سوپروایزری را به‌صورت احتمالی و زمان‌دار در رباتیک جمعی پیاده‌سازی کرده، به

[1].Regular Languages
[2].Generator
[3].Finite State Machine

کار گرفته است. در بخش قبل، SCT تشریح شد. در این بخش، به مراحل مورد نیاز برای محاسبه سوپروایزر احتمالی و زمان‌دار و تولید خودکار کد کنترلی پرداخته شده است. در پایان این بخش، لایه‌های نرم‌افزاری مورد نیاز برای پیاده‌سازی ptSCT در سکوی آرگوس مورداشاره قرار گرفته‌اند. در طراحی ایدئال یک سیستم، هیچ حالتی نباید دارای بیش از یک رخداد قابل‌کنترل فعال باشد. در این صورت، به‌ازای هر ورودی، تنها یک خروجی داریم. در طراحی واقعی، چنین چیزی همیشه برقرار نیست؛ زیرا مشخصه‌های کنترلی لزوماً همه ترکیبات ممکن رخدادها را محدود نمی‌کنند [۱۵]. ماشین حالت متناهی مورد استفاده در مولدها دارای محدودیت‌هایی در طراحی است. برای نمونه، عدم‌پشتیبانی از رخدادهای احتمالی از تعیین اولویت بین رخدادها جلوگیری می‌کند؛ برای مثال، ممکن است در یک مسئله نیاز به اعمال اولویت بیشتر به حرکت روبه‌جلو نسبت به چرخش به طرفین وجود داشته باشد یا چرخش به یکی از طرفین، بیشتر از دیگری رخ دهد. با استفاده از رخدادهای احتمالی پیشنهادشده در این مقاله، می‌توان در مرحله طراحی و به‌صورت صوری، عدم‌قطعیت رخدادها را در نظر گرفت و اعمال کرد.

محدودیت دوم ماشین‌های حالت متناهی، عدم پشتیبانی از زمان است. با اضافه کردن زمان به رخدادها، می‌توان دستهٔ بزرگ‌تری از مسائل را حل کرد. در تئوری کنترل سوپروایزری زمان‌دار که پیش از این معرفی شده است، یک ساعت سراسری در سیستم وجود دارد که مستقل از کنترل کننده‌ها عمل می‌کند [۱۸]. برای هر گذار، دو زمان قابل‌تعیین است؛ کران پایین و کران بالا. کران پایین، تعیین‌کنندهٔ مقدار تأخیر اعمال‌شده به زمان جاری سیستم قبل از اجرای رخداد قابل‌کنترل مربوط به گذار موردنظر است. رخداد زمان‌دار نمی‌تواند قبل از فرارسیدن کران پایین زمانش، فعال شود. کران بالا، زمان انقضای رخداد را مشخص می‌کند. یک رخداد پس از رسیدن به کران بالای زمانش، قابل فعال شدن نیست (منقضی می‌شود). احتمال و زمان، تنها مختص به رخدادهای قابل‌کنترل هستند.

در ادامه، به تعریف صوری تئوری کنترل سوپروایزری احتمالی و زمان‌دار پیشنهادی (ptSCT) پرداخته شده است. مولّد مربوط به ptSCT به‌صورت $(Q, \Sigma, \delta, q_0, Q_m, P, T)$ تعریف می‌شود. P بیانگر احتمال رویداد رخدادها در هر گذار و T بیانگر کران پایین و بالای زمان مربوط به رخدادها هستند: $P: Q \times \Sigma \rightarrow [0,1]$ و $T: Q \times \Sigma \rightarrow [l,u]$. l کران پایین و u کران بالای گذار را نشان می‌دهد: $l \in \mathbb{N}, u \in \mathbb{N} \cup \{\infty\}$. مجموعه رخدادهای ptSCT شامل یک رخداد جدید، یعنی تیک ساعت سراسری سیستم است؛ بنابراین $\Sigma = \Sigma_{SCT} \cup \{tick\}$. این رخداد مستقل از کنترل‌کننده‌ها در هر واحد زمانی رخ می‌دهد. درضمن، مجموع احتمالات خروجی از هر حالت برابر با یک است: $\forall q \in Q, \sum_{e \in \Sigma} p(q,e) = 1$. استفاده از کران بالا، علاوه بر افزایش توانایی حل مسائل دشوارتر، پیچیدگی را نیز افزایش می‌دهد. ازاین‌رو، با درنظرگرفتن $u = \infty$ برای همه گذارها در ptSCT، با پذیرش کاهش نسبی توان حل مسئله، پیچیدگی تا حد بسیار زیادی کاهش پیدا خواهد کرد. SCT یک حالت خاص از ptSCT است که به‌ترتیب با صفر و یک قرار دادن کران پایین و احتمال همه رخدادها حاصل می‌شود.

۳-۱- محاسبهٔ خودکار سوپروایزر

برای به‌دست‌آوردن سوپروایزر، مولّدهای مربوط به مدل رفتار آزاد و مشخصه‌های کنترلی با هم ترکیب می‌شوند. در فرایند ترکیب، مولّد مربوط به مدل رفتار آزاد با توجه به مشخصه‌های کنترلی محدود می‌شود. درواقع، ربات تنها قادر به اجرای اعمالی است که توسط مشخصه‌های کنترلی مجاز اعلام شده باشد. در این مقاله، سیستمی طراحی شده که همه مراحل موردنیاز برای محاسبه سوپروایزر احتمالی و زمان‌دار را به‌صورت خودکار انجام می‌دهد. شکل (۳) طراحی متنی معادل با مشخصه کنترلی شکل (۲) را که توسط سیستم پیشنهادی پذیرفته می‌شود، به نمایش گذاشته است. در این طراحی، تمام جزئیات مربوط به حالت‌ها، رخدادها و گذارها در قالب XML توصیف می‌شود.

```
<automata name="E1" type="specification">
    <states>
        <state name="p1" i="1" m="1"/>
        <state name="p2" m="1"/>
    </states>

    <events>
        <event con="1" name="move"/>
        <event con="1" name="stop"/>
        <event con="0" name="active"/>
        <event con="0" name="inactive"/>
    </events>

    <transitions>
        <transition event="move" src="p1" dst="p1"/>
        <transition event="stop" src="p2" dst="p2"/>
        <transition event="active" src="p1" dst="p2"/>
        <transition event="inactive" src="p2" dst="p1"/>
    </transitions>
</automata>
```

شکل ۳: طراحی متنی مشخصه کنترلی نوار نقاله (شکل ۲) که توسط سیستم پیشنهادی به‌عنوان ورودی دریافت می‌شود

برای ترکیب دو مولّد G_a و G_b با الفبای $\sum_i,\ i \in \{a,b\}$ از عملگر ترکیب موازی[1] به‌صورت نشان‌داده‌شده در فرمول ۱ استفاده می‌شود. این عملگر با نماد $\|$ نمایش داده شده است. تابع انتقال $\delta_{a\|b}$ مطابق با فرمول ۲

محاسبه می‌شود.
این لحظه، موجب اشغال فضای بیشتر در حافظه ربات و افزایش زمان پیاده‌سازی لایه رویه‌های عملیاتی نیز خواهد شد.

$$G_a \parallel G_b = (Q_a \times Q_b, \sum_a \cup \sum_b, \delta_{a\|b}, (q_{0_a}, q_{0_b}), Q_{m_a} \times Q_{m_b}, P_{a\|b}, T_{a\|b}) \quad (۱)$$

$$\delta_{a\|b}((q_a,q_b),e) = \begin{cases} (\delta_a(q_a,e),\delta_b(q_b,e)) & if\ \delta_a(q_a,e)\ and\ \delta_b(q_b,e)\ defined \\ (\delta_a(q_a,e),q_b) & if\ \delta_a(q_a,e)\ defined\ and\ e \notin \sum_b \\ (q_a,\delta_b(q_b,e)) & if\ \delta_b(q_b,e)\ defined\ and\ e \notin \sum_a \\ undefined & otherwise \end{cases} \quad (۲)$$

$$P_{a\|b}((q_a,q_b),e_c) = \begin{cases} P_a(q_a,e_c) \times P_b(q_b,e_c) & if\ \delta_a(q_a,e)\ and\ \delta_b(q_b,e)\ defined \\ P_a(q_a,e_c) & if\ \delta_a(q_a,e)\ defined\ and\ e \notin \sum_b \\ P_b(q_b,e_c) & if\ \delta_b(q_b,e)\ defined\ and\ e \notin \sum_a \\ 0 & otherwise \end{cases} \quad (۳)$$

$$T_{a\|b}((q_a,q_b),e_c) = \begin{cases} max(T_a(q_a,e_c),T_b(q_b,e_c)) & if\ \delta_a(q_a,e)\ and\ \delta_b(q_b,e)\ defined \\ T_a(q_a,e_c) & if\ \delta_a(q_a,e)\ defined\ and\ e \notin \sum_b \\ T_b(q_b,e_c) & if\ \delta_b(q_b,e)\ defined\ and\ e \notin \sum_a \\ 0 & otherwise \end{cases} \quad (۴)$$

فرمول ۲ اجازه می‌دهد رخدادهایی که بین دو مولّد مشترک نیستند، به‌صورت غیرهم‌زمان عمل کنند. نحوه محاسبه احتمال و زمان گذارهای مربوط به رخدادهای قابل‌کنترل، به‌ترتیب در فرمول‌های ۳ و ۴ ارائه شده‌اند.
با ترکیب موازی مدل رفتار آزاد G و مشخصه کنترلی E، مولّد زبان هدف (K) یا سوپروایزر حاصل می‌شود:

$$K = G \parallel E \quad (۵)$$

در این مرحله زبان $L(K)$ لزوماً کنترل‌شونده نیست. زبان K با الفبای $\sum$ و مجموعه رخدادهای غیرقابل‌کنترل $\sum_u \subseteq \sum$ در صورتی کنترل‌شونده است که [۱۹]:

$$\forall s \in \overline{L(K)}, \forall e_u \in \sum_u, se_u \in L(G) \rightarrow se_u \in \overline{L(K)} \quad (۶)$$

نماد $\bar{L}$ بیانگر زبان پیشوند بسته‌شدهٔ زبان L است و به‌صورت زیر تعریف می‌شود:

$$\bar{L} = \{s \in \sum^* : \exists t \in \sum^* \wedge st \in L\} \quad (۷)$$

به بیانی دیگر، اگر s پیشوند یکی از رشته‌های تولیدشده توسط مولد K باشد و e_u یک رخداد غیرقابل‌کنترل که از نظر فیزیکی امکان رخ‌دادن آن وجود دارد $(se_u \in L(G))$، زبان $L(K)$ در صورتی کنترل‌شونده است که رشته se_u نیز پیشوند یکی از رشته‌های تولیدشده توسط مدل K باشد. درصورتی‌که اینچنین نباشد، زبان $L(K)$ غیرقابل‌کنترل بوده، حالتی که در آن رخداد غیرقابل‌کنترل e_u رخ داده است، حالت بد نامیده می‌شود.
برای استخراج زیرمجموعه‌ای از زبان $L(K)$ که کنترل‌شونده است، باید همه حالت‌های بد و سایر حالت‌هایی که توسط رخدادهای غیرقابل‌کنترل به حالت‌های بد می‌رسند، حذف شوند. به زبان حاصل، بزرگ‌ترین زیرمجموعه کنترل‌شونده زبان $L(K)$ گفته می‌شود [۲۰].
پس از به‌دست‌آوردن زبان کنترل‌شونده، نوبت به حذف بن‌بست‌ها می‌رسد. درصورتی‌که زبان دارای یکی از دو نوع

[1].Parallel Composition

بن‌بست مرده[1] و زنده[2] باشد، بلاک‌شونده است. برای حذف بن‌بست‌ها، باید حالت‌های غیرقابل‌دسترس از حالت ابتدایی حذف شوند. علاوه براین، حالت‌هایی که مسیری بین آن‌ها و حداقل یکی از حالت‌های علامت‌دار وجود ندارد نیز باید حذف شوند. حذف این دو مجموعه از حالت‌ها، بلاک‌شونده نبودن زبان $L(K)$ را تضمین می‌کند [۲۰]. در این مرحله، به نرمال‌سازی احتمالات مربوط به هر حالت پرداخته می‌شود، به شکلی که مجموع آن‌ها برابر با یک باشد. سوپروایزر نهایی به‌صورت نشان‌داده‌شده در فرمول ۸ به دست می‌آید. عملگر $SupC$، با توجه به مدل رفتار آزاد G، ابتدا حالت‌های بد و سپس حالت‌های غیرقابل‌دسترس را از مولّد K حذف کرده، سپس به نرمال‌سازی احتمالات می‌پردازد. مولّد نهایی، حاصل S کنترل‌شونده و بلاک‌نشونده است.

$$S = SupC(K, G) \qquad (۸)$$

۳-۲- پیاده‌سازی ptSCT بر روی سکوی آرگوس

برای فراهم‌آوردن امکان اجرای سوپروایزر در سکوی آرگوس، نیاز به پیاده‌سازی سه لایه نرم‌افزاری بر روی ربات است (شکل۴). منطق لایه‌ها مشابه روند طی‌شده در [۹] خواهد بود، هرچند پیاده‌سازی صورت‌گرفته کاملاً متفاوت بوده و جامع‌تر است. در پایین‌ترین سطح، لایه رویه‌های عملیاتی[3] قرار می‌گیرد. وظیفه این لایه، تبدیل سیگنال‌ها به رخدادها و برعکس است. درواقع، این لایه، لایه سخت‌افزاری را از سایر لایه‌های نرم‌افزاری پنهان می‌کند. لایه دوم، اجراکننده ماشین حالت[4] است. این لایه یک ماشین حالت را دریافت و در هر گام ربات، یک مرحله از ماشین حالت را اجرا می‌کند. در بالاترین لایه، سوپروایزر قرار دارد که از یک یا چند ماشین حالت و تعدادی قطعه کد تشکیل شده است. هر تکه کد در زمان رویداد یک رخداد قابل‌کنترل، اجرا می‌شود. پشتیبانی از رخدادهای زمان‌دار، از وظایف این لایه است. نرمال‌سازی احتمال رخدادها در بین سوپروایزرها نیز از وظایف دیگر این لایه است. نرمال‌سازی احتمالات در دو سطح انجام می‌شود؛ سطح اول در مرحله محاسبه سوپروایزر صورت می‌گیرد و از یک بودن مجموع احتمالات هر حالت اطمینان حاصل می‌کند. سطح دوم به نرمال‌سازی رویدادهای یکسان بین سوپروایزرهای متفاوت می‌پردازد.

۴- پیاده‌سازی وظیفه دوری از موانع

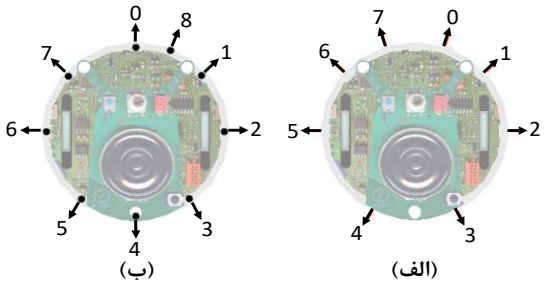
در وظیفه دوری از موانع، ربات‌ها سعی می‌کنند از سایر ربات‌ها و موانع موجود در محیط، دوری کرده، از برخورد با آن‌ها جلوگیری کنند. روش‌های مختلفی برای پیاده‌سازی این وظیفه ارائه شده است [۹, ۲۱, ۲۲]. برخلاف روش‌های قبلی، تمرکز طراحی پیشنهادشده بر روی تأثیر وجود احتمال و زمان در طراحی ساده‌تر و قدرتمندتر است. این وظیفه معمولاً در وظیفه‌های بزرگ‌تر و پیچیده‌تر مانند تجمع ربات‌ها و جست‌وجوی هدف در محیط، مورداستفاده قرار می‌گیرد.

شکل ۴: لایه‌های پیاده‌سازی‌شده بر روی لایه سخت‌افزار توسط روش پیشنهادی در سکوی آرگوس

روش پیشنهادی از ربات ایپاک[5] [۲۳] برای انجام آزمایش‌ها بهره برده است. این ربات می‌تواند از دوربین و یا سنسور مجاورت‌سنج برای تشخیص موانع در محیط استفاده کند. دوربین VGA این ربات، تصاویری با وضوح ۶۴۰ * ۴۸۰ برمی‌گرداند. حافظه ایپاک به‌قدری کوچک است که امکان ذخیره‌سازی یک تصویر را با این ابعاد ندارد؛ به همین دلیل، برای نمونه مرجع [۲۴] پس از کاهش ابعاد تصویر، تنها یک ستون به ارتفاع ۱۵ و عرض ۱ پیکسل را استخراج کرده، با رأی‌گیری از رنگ پیکسل‌های موجود در این ستون، درمورد وجود یا عدم وجود شیء در مقابل ربات، تصمیم می‌گیرد. در مقالهٔ حاضر، از سنسور مجاورت‌سنج به‌جای دوربین بهره برده شده است. محدودیت استفاده از سنسور، بسیار بیشتر از دوربین است؛ زیرا بازه دید دوربین ایپاک در حدود ۱۴۰ سانتی‌متر است، درصورتی‌که سنسور مجاورت‌سنج دارای برد ۸ سانتی‌متری است. علاوه براین، با استفاده از دوربین می‌توان وجود اشیای رنگی یا خاکستری را تشخیص داد، در حالی که سنسور مجاورت‌سنج تنها وجود اشیا در

[1].Deadlock
[2].Livelock
[3].Operational Procedures
[4].Automata Player
[5].E-Puck

مجاورت ربات را گزارش می‌کند و هیچ اطلاعاتی از نوع شیء مربوط در اختیار قرار نمی‌دهد. در طرف مقابل، هزینه استفاده از سنسور مجاورت‌سنج کمتر از دوربین است. ایپاک دارای هشت سنسور مجاورت است که در شکل (۵) (الف) نمایش داده شده‌اند. روش پیشنهادی از ترکیب سنسورهای شماره صفر و هفت برای تشخیص موانع در پیش‌رو بهره می‌برد. خروجی این سنسورها مقداری در بازه [۰-۱] است. صفر به‌معنی عدم وجود شیء و یک به‌معنی وجود شیء در نزدیک‌ترین فاصله ممکن به سنسور است. برای ترکیب دو سنسور شماره صفر و هفت، بیشینه مقدار خروجی آن‌ها مورداستفاده قرار گرفته است. در ضمن، طراحی سوپروایزر کاملاً بر اساس مراحل توضیح‌داده‌شده در بخش‌های قبل صورت می‌گیرد.



شکل ۵: (الف) مکان سنسورهای مجاورت‌سنج، (ب) مکان چراغ‌های LED ربات ایپاک که از نمای بالا نمایش داده شده‌اند [۱۷]

شکل (۶) (G_1 و G_2)، مدل‌های رفتار آزاد، مشخصهٔ کنترلی و سوپروایزرپیشنهادشده برای وظیفه دوری از موانع را نمایش داده است. مدل رفتار آزاد G_1 حرکت‌های ممکن ربات، یعنی حرکت مستقیم، چرخش به چپ و راست را توصیف می‌کند. وقتی رخداد چرخش به چپ اتفاق می‌افتد، تا زمان رویداد رخدادی دیگر، ربات به‌صورت تکراری به سمت چپ می‌چرخد. مدل رفتار آزاد G_2، رفتار اصلی ربات، یعنی دوری از موانع را توصیف می‌کند. ربات تا زمانی که مانعی حس نکند، به حرکت مستقیم خود ادامه می‌دهد. رخداد غیرقابل‌کنترل $seen\_obstacle$ درصورتی‌که مانعی در مقابل ربات باشد، مقدار یک و درغیراین‌صورت، مقدار صفر را برمی‌گرداند. حالت‌های ابتدایی با یک فلش بدون مبدأ و حالت‌های پایانی با یک دایره دوخطی نمایش داده شده‌اند. گذارهای دوخطی نشان‌دهندهٔ رخدادهای غیرقابل‌کنترل هستند. احتمال هر رخداد در کنار آن و بعد از علامت : قرار گرفته است.

مشخصه کنترلی E_1، تنها در صورتی چرخش به چپ یا راست را فعال[1] می‌کند که مانعی در مقابل ربات دیده شود. احتمال چرخش به چپ، دو برابر چرخش به راست تعیین شده است. این امر موجب رهایی سریع‌تر ربات از تنگناها می‌شود. در صورت چرخش ربات به یکی از طرفین، ابتدا به مقدار سه واحد زمانی از ساعت سیستم، انتظار صورت گرفته، سپس ربات به مسیر مستقیم ادامه می‌دهد. زمان هر رخداد در کنار آن و در داخل پرانتز نمایش داده شده است. رخدادهای بدون زمان، دارای زمان صفر هستند. حرکت مستقیم یا چرخش ربات، با تنظیم سرعت موتورهای ربات انجام می‌شوند. این عملیات و عملیات مربوط به سایر رخدادها در لایه رویه‌های عملیاتی پیاده‌سازی می‌شوند و مستقل از طراحی کنترل‌کننده‌ها هستند.

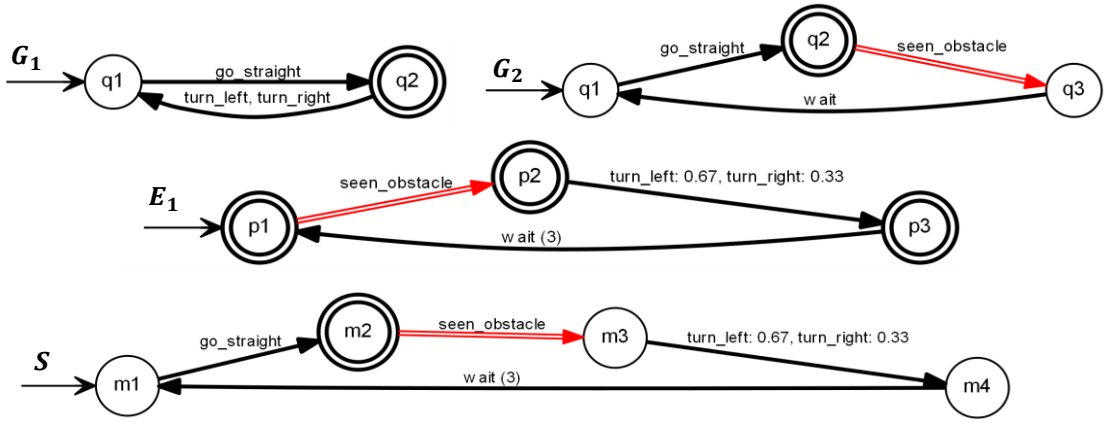
سوپروایزر S از مدل‌های رفتار آزاد G_1 و G_2 و مشخصهٔ کنترلی E_1، توسط سیستم پیشنهادی و با استفاده از ptSCT محاسبه شده است. برای اجرای استراتژی دوری از موانع در محیط شبیه‌سازی یا در دنیای واقعی، کافی است کد تولیدشده برای این سوپروایزر به‌عنوان نرم‌افزار کنترلی ربات‌ها مورداستفاده قرار گیرد.

در ادامه، برای مقایسه مزایای طراحی با استفاده از ptSCT نسبت به SCT، همین وظیفه با استفاده از SCT و بدون به‌کارگیری احتمالات و زمان پیاده‌سازی شده است. شکل (۷) مدل‌های رفتار آزاد طراحی‌شده را نمایش داده است. مدل G_3، مدل جدیدی است که در این شکل دیده می‌شود و وظیفه ایجاد تأخیر را برعهده دارد. عدم وجود رخدادهای زمان‌دار، استفاده از چنین مدلی را اجبار کرده است. مدل G_3 با رویداد رخداد $start\_timer$، شمارش را آغاز کرده، به کمک شش رویداد غیرقابل کنترل، سه گام از اجرای سیستم را می‌شمارد. رخدادهای $wait\_1s$، $wait\_2s$ و $wait\_3s$ به‌ترتیب در گام اول، دوم و سوم از ساعت سیستم (پس از رسیدن به حالت مربوط) فعال خواهند شد. رخداد inc_timer یک واحد به شمارنده اضافه می‌کند. رخداد $timer\_eq\_i$ زمانی که شمارنده برابر با i باشد، فعال می‌شود، به‌طوری‌که $i \in \{1,2,3\}$. رخداد $timer\_lw\_i$ زمانی که شمارنده کوچک‌تر از i باشد، فعال می‌شود.

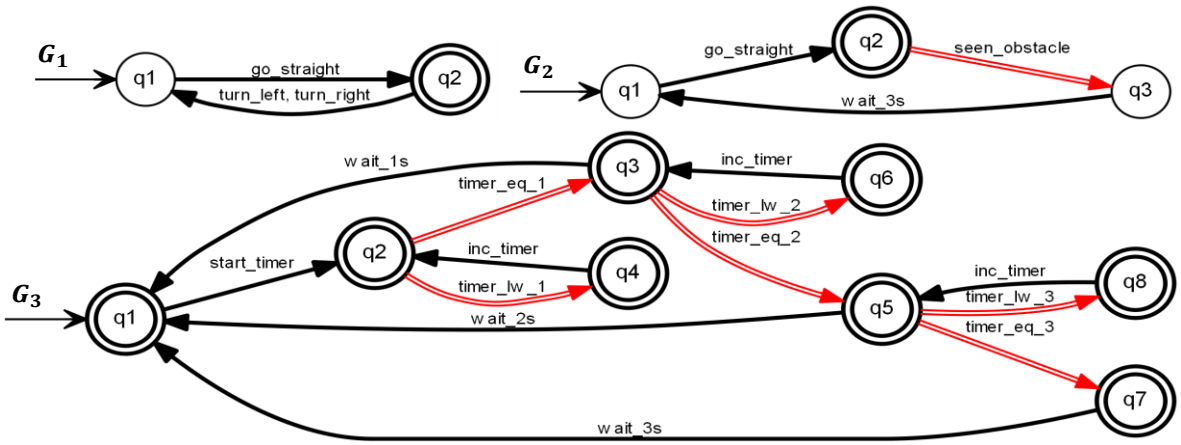
[1].Enabled

مشخصه کنترلی E_1 در شکل (۸) تقریباً مشابه با مشخصه کنترلی معادل در شکل (۶) است، با این تفاوت که رخداد $start\_timer$ به آن اضافه شده، زمان رخداد $wait$ در آن وجود ندارد. علاوه براین، مشاهده می‌شود که بدون استفاده از احتمال، چرخش به چپ و راست از یک شانس برخوردارند. بنابراین، احتمال به دام افتادن ربات در تنگاها برای مدت طولانی وجود دارد که قابل‌اجتناب نیست.؛ برای مثال، ربات در مواجهه با مانع، ممکن است یک مرتبه به چپ و سپس به راست بچرخد و به این ترتیب تا مدت طولانی نتواند از مانع موردنظر عبور کند.

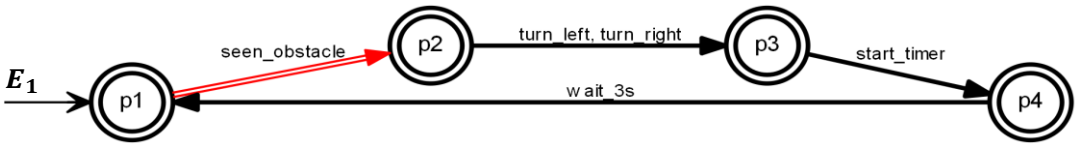
سوپروایزر محاسبه‌شده در شکل (۹) به‌دلیل وجود شمارنده در آن، بسیار بزرگ‌تر از معادل محاسبه‌شده در شکل (۶) است. این امر علاوه بر اشکالات برشمرده شده تا این لحظه، موجب اشغال فضای بیشتر در حافظه ربات و افزایش زمان پیاده‌سازی لایه رویه‌های عملیاتی نیز خواهد شد.



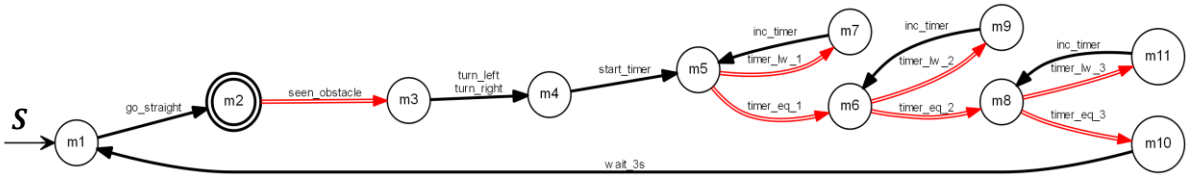
شکل ۶: مدل‌های رفتار آزاد (G_1 و G_2)، مشخصه کنترلی (E_1) و سوپروایزر (S) مربوط به وظیفه دوری از موانع با استفاده از ptSCT



شکل ۷: مدل‌های رفتار آزاد مربوط به وظیفه دوری از موانع با استفاده از SCT



شکل ۸: مشخصهٔ کنترلی مربوط به وظیفه دوری از موانع با استفاده از SCT



شکل ۹: سوپروایزر مربوط به وظیفه دوری از موانع با استفاده از SCT

۵- پیاده‌سازی وظیفه همگام‌سازی با استفاده از ptSCT

همگام‌سازی از وظایف پرکاربرد در رباتیک جمعی است؛ زیرا گام اول در بسیاری از وظایف پیچیده‌تر را تشکیل می‌دهد. برای نمونه، از همگام‌سازی می‌توان برای تشخیص ربات‌های دچارمشکل‌شده استفاده کرد [۲۵] یا باکتری‌ها برای تشخیص اندازه جمعیت از همگام‌سازی بهره می‌برند [۲۶]. در وظیفه پیاده‌سازی‌شده در این بخش، هر ربات دارای یک شمارندهٔ داخلی است. این شمارنده از یک مقدار تصادفی، آغاز به شمارش کرده، پس از رسیدن به یک مقدار مشخص، شمارش را از صفر آغاز می‌کند. بنابراین، سقف شمارش برای همه ربات‌ها یکسان، ولی مقدار آغازین برای هر ربات متفاوت است. ربات‌ها باید بتوانند پس از مدتی، شمارنده‌هایشان را با یکدیگر همگام کنند. این مسئله، همان همگام‌سازی فازها[1] است. هر ربات وقتی به سقف شمارشش می‌رسد، با روشن کردن چراغ‌های LED خود برای مدت معین، به سایرین علامت می‌دهد. هر رباتی که چراغ سایر ربات‌ها را می‌بیند، با استفاده از فرمول ۹ مقدار شمارندهٔ خود را تغییر می‌دهد.

$$counter = counter + \alpha \,.\, counter \qquad (۹)$$

ثابت α مقداری در بازه [۰-۱] بوده، برای همه ربات‌ها یکسان است. در این وظیفه از دو ویژگی سخت‌افزاری ربات ایپاک استفاده شده است. چراغ‌های LED و دوربین ۳۶۰ درجه. هر ربات به‌وسیلهٔ دوربینش، روشن بودن چراغ سایرین را تشخیص داده، شمارنده خود را تغییر می‌دهد. مکان نه چراغ LED ربات ایپاک در شکل (۵-ب) نمایش داده شده‌اند. همه این چراغ‌ها به‌صورت هم‌زمان روشن و خاموش می‌شوند.

شکل (۱۰)، مدل‌های رفتار آزاد و مشخصه‌های کنترلی پیشنهادشده برای وظیفه همگام‌سازی را نمایش داده است. مدل رفتار آزاد G_1 به قابلیت ربات در روشن و خاموش کردن چراغ‌های LED اشاره دارد. رخدادهای قابل‌کنترل led_on و led_off به‌ترتیب چراغ‌های ربات را روشن و خاموش می‌کنند. مدل G_2 شکل عملکرد شمارنده ربات را توصیف می‌کند. ربات تا زمانی که به سقف شمارش نرسیده است (رخداد غیرقابل‌کنترل $reached\_top$)، شمارنده‌اش را افزایش می‌دهد (رخداد $inc\_counter$)؛ درغیراین‌صورت، شمارنده را صفر خواهد کرد (رخداد $reset$). بعد از هر شمارش، رخداد $timeout$ اتفاق خواهد افتاد که به اندازه یک واحد زمانی از ساعت سیستم تأخیر اعمال می‌کند. این امر موجب می‌شود در هر گام از اجرای سیستم، شمارنده تنها یک واحد افزایش یابد. مدل G_3 عملکرد ربات را در ارتباط با سایرین توصیف می‌کند. ربات تا زمانی که روشنایی در اطرف خود مشاهده نکرده است، به شمارش ادامه می‌دهد. در صورت مشاهده روشنایی، شمارنده خود را توسط رخداد $sync$ همگام خواهد کرد.

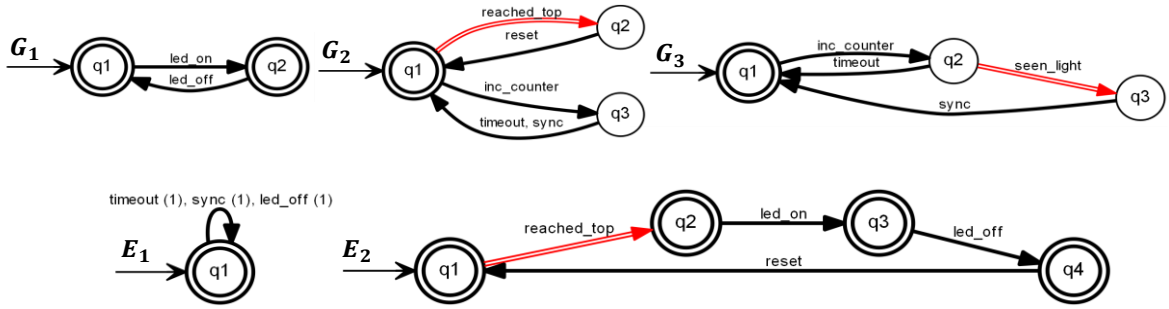
مشخصهٔ کنترلی E_2، عملکرد ربات را در زمان رسیدن شمارنده به سقف شمارش تعیین می‌کند؛ به این شکل که ابتدا چراغ LED ربات روشن شده، پس از مدتی خاموش و سپس، شمارنده صفر می‌شود. مشخصه کنترلی E_1، زمان رخدادهای زمان‌دار را تعیین کرده است. اجرای این رخدادها باید با یک واحد زمانی تأخیر صورت گیرد.

شکل (۱۱)، سوپروایزر به‌دست‌آمده از ترکیب مدل‌های رفتار آزاد و مشخصه‌های کنترلی را نشان می‌دهد. همان طور که در شکل مشخص است، منطق محاسبه‌شده کاملاً صحیح است. علاوه براین، زمان رخدادهای زمان‌دار نیز به‌درستی اعمال شده است.

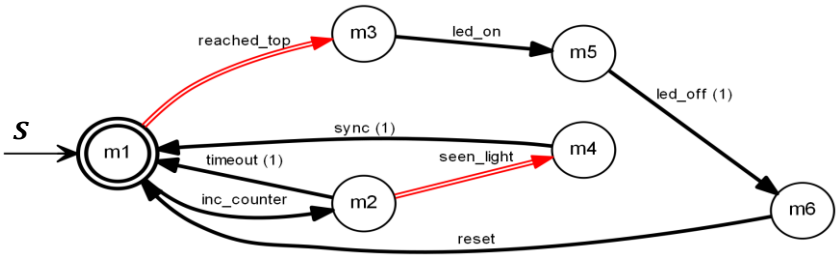
در ادامه، مشابه بخش قبل، همین وظیفه با استفاده از SCT و بدون استفاده از احتمالات و زمان پیاده‌سازی شده است. شکل (۱۲) مدل‌های رفتار آزاد طراحی‌شده را نمایش داده است. مدل‌های G_4 تا G_6، مدل‌های جدیدی هستند که به طراحی قبلی اضافه شده‌اند و وظیفه ایجاد تأخیر را برعهده دارد. عدم وجود رخدادهای زمان‌دار، استفاده از چنین مدل‌هایی را اجبار کرده است. شمارنده iام با رویداد رخداد $start\_timer\_i$، شمارش را آغاز کرده، به کمک دو رویداد غیرقابل کنترل $timer\_i\_lw\_1$ و $timer\_i\_eq\_1$، یک گام از اجرای سیستم را می‌شمارد. در پایان شمارش، رخداد end_timer_i فعال خواهد شد. رویداد inc_timer_i یک واحد به شمارنده iام اضافه می‌کند. رخداد $timer\_i\_eq\_1$ زمانی که شمارنده iام برابر با یک باشد، فعال شده، رخداد $timer\_i\_lw\_1$ زمانی

[1].Phase Synchronization

که شمارنده iام کوچک‌تر از یک باشد، فعال می‌شود.



شکل ۱۰: مدل‌های رفتار آزاد (G_1، G_2 و G_3) و مشخصه‌های کنترلی (E_1) مربوط به وظیفه همگام‌سازی با استفاده از ptSCT

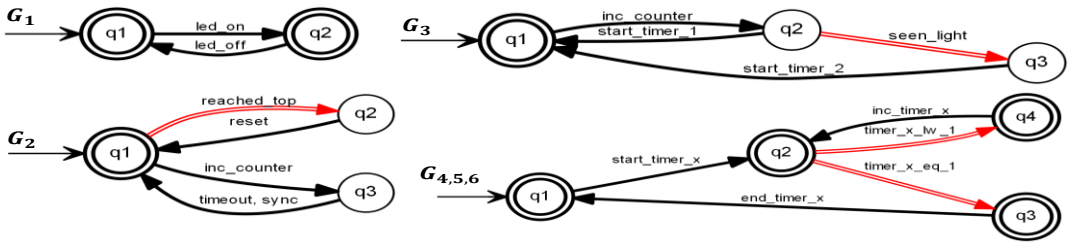


شکل ۱۱: سوپروایزر مربوط به وظیفه همگام‌سازی با استفاده از ptSCT

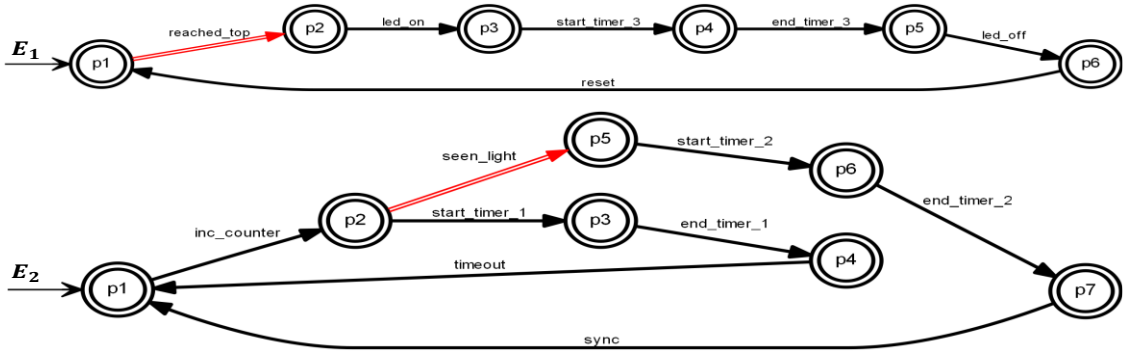
در پایان شمارش، رخداد end_timer_i فعال خواهد شد. رویداد inc_timer_i یک واحد به شمارنده iام اضافه می‌کند. رخداد $timer\_i\_eq\_1$ زمانی که شمارنده iام برابر با یک باشد، فعال شده، رخداد $timer\_i\_lw\_1$ زمانی که شمارنده iام کوچک‌تر از یک باشد، فعال می‌شود.

شکل (۱۳) مشخصه‌های کنترلی جدید را نمایش داده است. وظیفه هر دو مشخصهٔ کنترلی E_1 و E_2، اضافه کردن تأخیرهای موردنیاز قبل از اجرای رویدادهایی است که نیاز به تأخیر دارند؛ یعنی سه رویداد $timeout$، $sync$ و led_off.

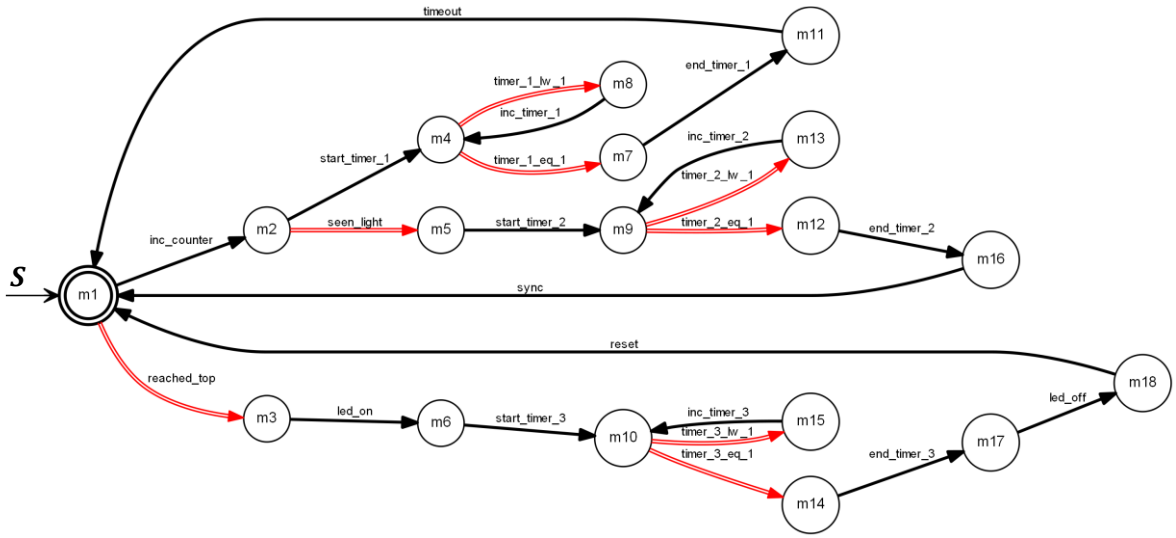
سوپروایزر محاسبه‌شده در شکل (۱۴) به‌دلیل وجود سه شمارنده در آن، بسیار بزرگ‌تر از معادل محاسبه‌شده در شکل (۱۱) است. این امر علاوه بر مشکلات برشمرده‌شده در بخش قبل، موجب افزایش دشواری در تحلیل و اشکال‌یابی طراحی انجام شده نیز خواهد شد.



شکل ۱۲: مدل‌های رفتار آزاد (G_1 تاG_6) مربوط به وظیفه همگام‌سازی با استفاده از SCT:$x \in \{1, 2, 3\}$



شکل ۱۳: مشخصه‌های کنترلی (E_1 و E_2) مربوط به وظیفه همگام‌سازی با استفاده از SCT



شکل ۱۴: سوپروایزر مربوط به وظیفه همگام‌سازی با استفاده از SCT

۶- نتایج آزمایش‌ها

تمامی آزمایش‌های انجام‌شده در این مقاله بر روی سکوی آرگوس[۱۶] صورت گرفته است. سکوی آرگوس یک شبیه‌ساز برای سیستم‌های چندرباته است که تأکیدی ویژه بر مقیاس جمعیت ربات‌ها با حفظ سرعت دارد. یکی از قابلیت‌های منحصربه‌فرد آرگوس، امکان کامپایل کد کنترلی برای ربات‌های واقعی است؛ بنابراین، نیازی به پیاده‌سازی دو برنامه متفاوت، یکی به هدف شبیه‌سازی و یکی به هدف ربات واقعی وجود ندارد. آرگوس به زبان سی‌پلاس‌پلاس نوشته شده، به‌صورت متن‌باز در دسترس محققان قرار دارد.

ابعاد محیط مورداستفاده برای انجام همه آزمایش‌ها برابر با ۳۰۰ در ۳۰۰ سانتی‌متر و ارتفاع نیم متر است. برای افزایش سرعت شبیه‌سازی، در هر ثانیه، ۱۰ دور از الگوریتم اجرا می‌شود. دوربین در ارتفاع دو و نیم متری از محیط شبیه‌سازی قرار داده شده، در زمان‌های مشخص، تصاویری تهیه می‌کند. اجرای سوپروایزرهای به‌دست‌آمده از روش SCT و ptSCT، نتیجه‌ای یکسان در برخواهند داشت. بنابراین، نمایش نتیجه اجرای یکی از آن‌ها کفایت می‌کند. تمرکز اصلی این مقاله بر روی روش طراحی سوپروایزر است، نه نتایج اجرای سیستم. ازاین‌رو، آزمایش‌های انتخاب‌شده از وظایف پایه‌ای در رباتیک جمعی هستند و نتایج آن‌ها به سادگی قابل‌تحلیل است. به همین دلیل، برای نمایش بخش‌های جدید روش پیشنهادی (دو فاکتور زمان و احتمال)، مراحل طراحی صوری وظایف با روش پیشنهادی و روش پایه (SCT) مقایسه شده‌اند تا مزیت‌های روش پیشنهادی، قابل‌لمس باشد.

۶-۱- شبیه‌سازی وظیفه دوری از موانع

برای شبیه‌سازی‌ها وظیفه دوری از موانع از ۵۰ ربات ایپاک در محیطی با موانع متعدد استفاده شد. به ورودی سنسور مجاورت‌سنج، نویز گوسی با انحراف معیار ۰٫۳ اعمال شده است تا به رفتار دنیای واقعی نزدیک‌تر شود. نویز مشابهی به عملکرد موتورهای ربات نیز اعمال شد [۲۷،۲۸]. مقدار ثابت α از فرمول ۹ نیز برابر با ۰/۱ تعیین شده است.

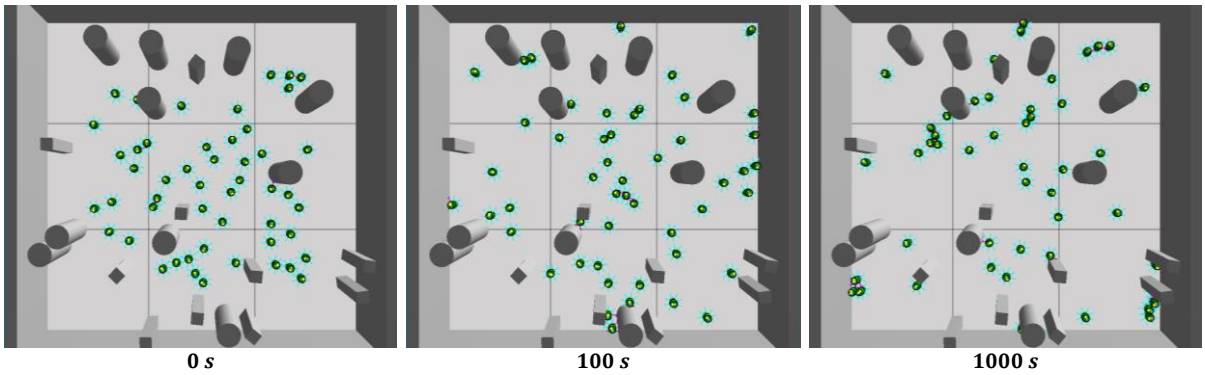
نتیجه آزمایش انجام‌شده در شکل (۱۵) نمایش داده شده که نشان از عملکرد صحیح سوپروایزر محاسبه‌شده دارد. هیچ رباتی توسط شیء یا رباتی دیگر متوقف نشده؛ چنان‌که در شکل نیز می‌بینیم، موقعیت ربات‌ها در زمان‌های مختلف به‌درستی تغییر کرده است.

۶-۲- شبیه‌سازی وظیفه همگام‌سازی

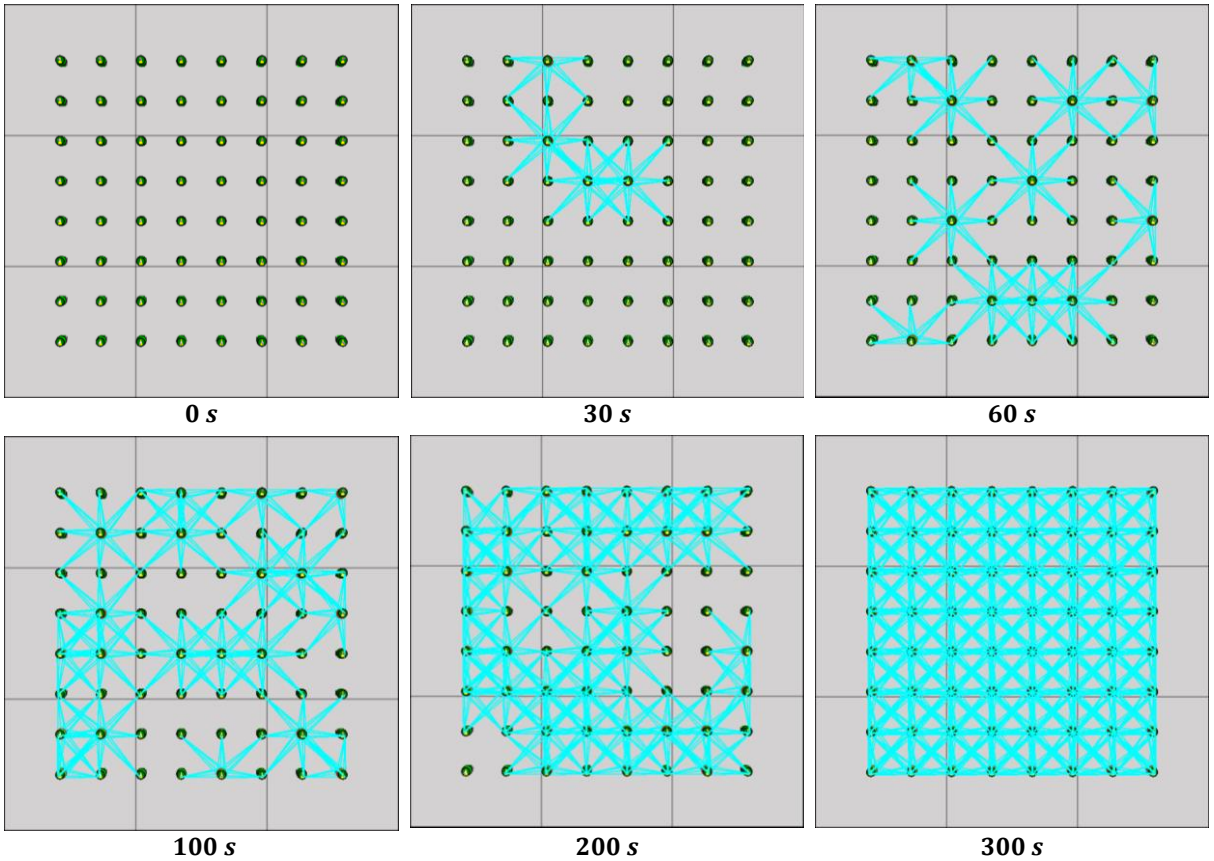
برای شبیه‌سازی وظیفه همگام‌سازی از ۶۴ ربات ایپاک در محیطی همگن استفاده شده است. ربات‌ها به‌صورت یک شبکه ۸ در ۸ چیده شده، فاصلهٔ آن‌ها به گونه‌ای تعیین شده که بتوانند چراغ‌های یکدیگر را ببینند.

شکل (۱۶) چیدمان اولیه ربات‌ها و وضعیت همگام‌سازی ربات‌ها را در زمان‌های مختلف نمایش داده است. وقتی ربات در یک جهت از اطراف خود، روشنایی ببیند، یک خط در همان جهت در محیط شبیه‌ساز ترسیم می‌شود تا وضعیت همگام‌سازی ربات‌ها توسط ببیننده قابل‌تشخیص باشد. بنابراین در ثانیه ۳۰۰ از شروع اجرای سیستم، همه ربات‌ها همگام شده‌اند. این امر از درستی عملکرد سوپروایزر طراحی‌شده حکایت دارد.

در دنبالهٔ تصاویر ارائه‌شده در شکل (۱۶) ، روند همگام‌سازی ربات‌ها نیز قابل‌مشاهده است. ابتدا ربات‌های همسایه به‌تدریج همگام شده و دسته‌های کوچکِ همگام را تشکیل داده‌اند و سپس دسته‌های کوچک در هم ادغام شده، دسته‌های بزرگ‌تر همگام را ساخته‌اند که در نهایت، به همگام‌سازی کلیه ربات‌ها منجر شده است.



شکل ۱۵: دنباله تصاویر گرفته‌شده در زمان‌های صفر، ۱۰۰ و ۱۰۰۰ ثانیه از آزمایش دوری از موانع



شکل ۱۶: دنباله تصاویر گرفته‌شده در زمان‌های صفر، ۳۰، ۶۰، ۱۰۰، ۲۰۰ و ۳۰۰ ثانیه از آزمایش همگام‌سازی

شکل (۱۶) چیدمان اولیه ربات‌ها و وضعیت همگام‌سازی ربات‌ها را در زمان‌های مختلف نمایش داده است. وقتی ربات در یک جهت از اطراف خود، روشنایی ببیند، یک خط در همان جهت در محیط شبیه‌ساز ترسیم می‌شود تا وضعیت همگام‌سازی ربات‌ها توسط ببیننده قابل‌تشخیص باشد. بنابراین در ثانیه ۳۰۰ از شروع اجرای سیستم، همه ربات‌ها همگام شده‌اند. این امر از درستی عملکرد سوپروایزر طراحی‌شده حکایت دارد.

در دنبالهٔ تصاویر ارائه‌شده در شکل (۱۶) ، روند همگام‌سازی ربات‌ها نیز قابل‌مشاهده است. ابتدا ربات‌های همسایه به‌تدریج همگام شده و دسته‌های کوچکِ همگام را تشکیل داده‌اند و سپس دسته‌های کوچک در هم ادغام شده، دسته‌های بزرگ‌تر همگام را ساخته‌اند که در نهایت، به همگام‌سازی کلیه ربات‌ها منجر شده است.

۷- نتیجه‌گیری

در این مقاله، تئوری کنترل سوپروایزری احتمالی و زمان‌دار ptSCT برای طراحی سوپروایزر در رباتیک جمعی پیشنهاد شده است. برای این منظور، سیستمی پیاده‌سازی شده که کلیه عملگرهای ماشین حالت و ترکیب کنترل‌کننده‌ها را به‌صورت خودکار انجام می‌دهد. تمامی اعتبارسنجی‌های موردنیاز از قبیل بررسی عدم وجود بن‌بست، حذف حالت‌های بد، کمینه‌سازی ماشین حالت و تشخیص اشکالات غیرعمدی در طراحی، توسط این سیستم به‌صورت خودکار انجام می‌شود. روش پیشنهادی پس از محاسبه سوپروایزر احتمالی و زمان‌دار، کد کنترلی موردنیاز برای اجرای آن در سکوی آرگوس را تولید می‌کند. از آنجا که ptSCT به‌صورت کامل بر روی این سکو پیاده‌سازی شده، این کد بدون هیچ تغییری قابل‌اجرا در محیط شبیه‌سازی یا بر روی ربات‌های واقعی است. با استفاده از ptSCT می‌توان مسائل پیچیده‌تر و دشوارتر را پیاده‌سازی کرد. علاوه براین، می‌توان مسائلی را که با استفاده از SCT نیز قابل‌پیاده‌سازی هستند، با هزینه کمتر حل کرد. منظور از هزینه، دشواری طراحی و فضای اشغال‌شده از حافظه ربات است. چنان‌که دیدیم، در دو مسئله پیاده‌سازی‌شده، اندازه مدل‌های رفتار آزاد و مشخصه‌های کنترلی طراحی‌شده با استفاده از SCT بزرگ‌تر از ptSCT بود. طبیعتاً ترکیب این مدل‌ها و به‌دست‌آوردن سوپروایزر نهایی نیز دشوارتر خواهد بود. از طرفی به‌دلیل بزرگ‌تر بودن سوپروایزر به‌دست‌آمده توسط SCT، فضای اشغال‌شده از حافظه ربات نیز بیشتر خواهد بود. تحلیل و اشکال‌زدایی از یک ماشین حالت بزرگ‌تر نیز به دشواری مسئله می‌افزاید.

رباتیک جمعی دارای محدودیت‌های زیادی است که حل مسائل را در آن دشوار می‌سازد. قدرت محاسباتی و ارتباطی پایین و جمعیت بالای ربات‌ها ازجملهٔ این موارد است. هر دوی این موارد در طراحی‌های صورت‌گرفته کاملاً رعایت شده‌اند. تمامی مراحل طراحی به‌صورت صوری انجام و روند طراحی مرحله به مرحله توضیح داده شده است. نتایج آزمایش‌ها نشان از عملکرد درست طراحی‌های انجام‌شده دارند.

تقدیر و تشکر

با تشکر از جناب آقای دکتر محسن بیگلری که با نظرات و پیشنهادهایشان موجب ارتقای کیفیت این مقاله شدند.

مراجع

[1] E. Şahin, “Swarm Robotics: From Sources of Inspiration to Domains of Application”, International workshop on swarm robotics, pp. 10–20, Springer, Berlin, Heidelberg,2004.

[2] M. Brambilla, E. Ferrante, M. Birattari and M. Dorigo, "Swarm Robotics : A Review from the Swarm Engineering Perspective", Swarm Intelligence, Vol. 7, No. 1, 2013, pp. 1–41.

[3] S. Prasad and S. Rawool, "Swarm Robotics : Nature Inspired Systems", International Journal of Engineering Research and General Science, Vol. 4, No. 5, 2016, pp. 168–174.

[4] M. Dorigo and A.F. Roosevelt, "Swarm Robotics", Autonomous Robots, No. Special Issue, 2014.

[۵] ف. میرزائی و ع.ا. پویان، «مروری بر رباتیک جمعی و جایگاه آن در سیستم‌های چندرباته»، نشریه مهندسی برق و الکترونیک ایران، پذیرفته شده، ۱۳۹۶، صفحه ۱-۲۰.

[۶] ا. نیکوبین، ع. قدوسیان و م.ر. وزواری، (۱۳۹۶) «طراحی مسیر بهینه برای ربات کابلی معلق به‌وسیله میانیاب چندجمله‌ای درجه چهار و الگوریتم مثلث بهینه‌گر»، مجله مدل‌سازی در مهندسی، سال ۱۵، شماره ۴۸، ۱۳۹۶، صفحه ۳۱-۴۴.

[۷] س.ا. موسویان و س. حسینی، «طراحی پایدارترین حرکت ربات متحرک در مسیر مشخص»، مجله مدل‌سازی در مهندسی، سال ۱۱، شماره ۳۳، ۱۳۹۲، صفحه ۱-۱۴.

[۸] م.س. فومنی، م.م. خطیبی، م. مرادی و م.ک.م. آبادی، «تحلیل سینماتیکی-سینتیکی پیمایش مستقیم‌الخط ربات انسان‌نما»، مجله مدل‌سازی در مهندسی، سال ۸، شماره ۱۷، ۱۳۸۸، صفحه ۱۷-۲۵.

[9] Y.K. Lopes, S.M. Trenkwalder, A.B. Leal, T.J. Dodd and R. Groß, "Supervisory Control Theory Applied to Swarm Robotics", Swarm Intelligence, Vol. 10, No. 1, 2016, pp. 65–97.

[10] J.C. Knight, C.L. DeJong, M.S. Gibble and L.G. Nakano, "Why Are Formal Methods Not Used More Widely?", Fourth NASA formal methods workshop, pp. 1–12, NASA, 1997.

[11] M. Brambilla, M. Dorigo and M. Birattari, "Property-Driven Design for Robot Swarms : A Design Method Based on Prescriptive Modeling and Model Checking", ACM Transactions on Autonomous and Adaptive Systems, Vol. 9, No. 4, 2015, pp. 17.

[12] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni and M. Birattari, "AutoMoDe: A Novel Approach to the Automatic Design of Control Software for Robot Swarms", Swarm Intelligence, Vol. 8, No. 2, 2014, pp. 89–112.

[13] J. King, R.K. Pretty and R.G. Gosine, "Coordinated Execution of Tasks in a Multiagent Environment", IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, Vol. 33, No. 5, 2003, pp. 615–619.

[14] A.F. Winfield, J. Sa, M.-C. Fernández-Gago, C. Dixon and M. Fisher, "On Formal Specification of Emergent Behaviours in Swarm Robotic Systems", International journal of advanced robotic systems, Vol. 2, No. 4, 2005, p. 39.

[15] Y.K. Lopes, S.M. Trenkwalder, A.B. Leal, T.J. Dodd and R. Groß, "Probabilistic Supervisory Control Theory (pSCT) Applied to Swarm Robotics", Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, 2016, pp. 1395–1403.

[16] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L.M. Gambardella and M. Dorigo, "ARGoS: A Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems", Swarm Intelligence, Vol. 6, No. 4, 2012, pp. 271–295.

[۱۷] ف. میرزائی، ع.ا. پویان و س. فردوسی، (۱۳۹۶). "پیاده‌سازی و شبیه‌سازی استراتژی تجمع ربات‌ها با استفاده از تئوری کنترل سوپروایزری در رباتیک جمعی"، سومین کنفرانس پردازش سیگنال و سیستم‌های هوشمند، دانشگاه صنعتی شاهرود، ۱۳۹۶، صفحه ۵-۱۳.

[18] B.A. Brandin and W.M.Wonham, "The Supervisory Control of Timed DES", IEEE Transactions on Automatic Control, Vol. 39, No. 2, 1994, pp. 329–342.

[19] C.G. Cassandras and S. Lafortune, Introduction to discrete event systems, Springer Science & Business Media, 2009.

[20] P.J.Ramadge, W.M. Wonham, "Supervisory Control of a Class of Discrete Event Processes", SIAM journal on control and optimization, Vol. 25, No. 1, 1987, pp. 206–230.

[21] H. Rezaee and F. Abdollahi, "A Decentralized Cooperative Control Scheme with Obstacle Avoidance for a Team of Mobile Robots", IEEE Transactions on Industrial Electronics, Vol. 61, No. 1, 2014, pp. 347–354.

[22] K. Souhila, A. Karim, "Optical Flow Based Robot Obstacle Avoidance", International Journal of Advanced Robotic Systems, Vol. 4, No. 1, 2007, p. 2.

[23] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, J. Zufferey, D. Floreano and A. Martinoli, "The E-Puck, a Robot Designed for Education in Engineering", Proceedings of the 9th conference on autonomous robot systems and competitions, 2009, pp. 59–65.

[24] M. Gauci, J. Chen, W. Li, T.J. Dodd and R. Groß, "Self-Organized Aggregation without Computation", The International Journal of Robotics Research, Vol. 33, No. 8, 2014, pp. 1145–1161.

[25] C. Ampatzis, E. Tuci, V. Trianni, A.L. Christensen and M. Dorigo, "Evolving Self-Assembly in Autonomous

Homogeneous Robots: Experiments with Two Physical Robots", Artificial Life, Vol. 15, No. 4, 2009, pp. 465–484.

[26] B.L. Bassler, "How Bacteria Talk to Each Other: Regulation of Gene Expression by Quorum Sensing", Current opinion in microbiology, Vol. 2, No. 6, 1999, pp. 582–587.

[27] N. Jakobi, P. Husbands and I. Harvey, “Noise and the Reality Gab: The Use of Simulation in Evolutionary Robotics”, European Conference on Artificial Life, 1995, pp. 704–720.

[28] S. Koos, J. Mouret and S. Doncieux, "The Transferability Approach : Crossing the Reality Gap in Evolutionary Robotics To Cite This Version : The Transferability Approach : Crossing the Reality Gap in Evolutionary Robotics", IEEE Transactions on Evolutionary Computation, Vol. 17, No. 1, 2015, pp. 122–145.