دانشگاه تهران

دانشکده برق و کامپیوتر

رباتیک پیشرفته

گزارش فاز دوم پروژه نهایی

مکان یابی ربات e-Puck

علی نورمحمدی اصل

امیر خضرایی

تابستان ۱۳۹۴

مقدمه

در راستای انجام پروژه نهایی درس و پس از اتمام فاز اول آن به ارائه گزارشی از فاز دوم پروژه می‌پردازیم. در فاز دوم با استفاده از نتایج به دست آمده از فاز اول، به انجام مکان‌یابی ربات پرداخته شده است. همچنین مکان یابی ربات در صورت ربوده شدن آن (Kidnapping) نیز در طول پروژه انجام شد.

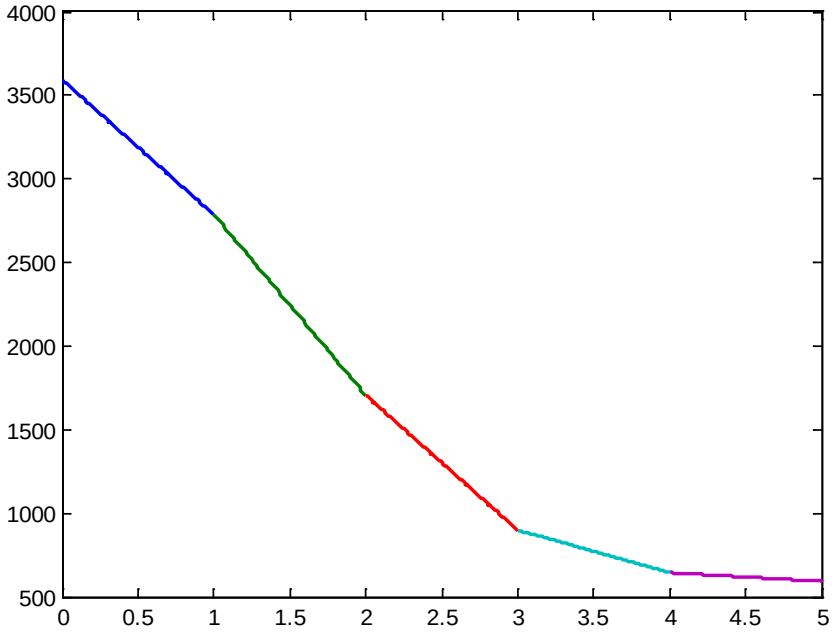
در این گزارش قصد داریم مجموعه مراحل انجام شده در این فاز را مورد بررسی قرار دهیم. البته در ابتدای گزارش به اصلاح قسمتی از فاز اول، که در طول انجام آزمایشات نتیجه گردید، می‌پردازیم.

۱- اصلاحات فاز اول

در فاز اول پروژه مدل سنسورهای ربات و مدل حرکت ربات e-Puck به‌دست آمد. قبل از بیان توضیحات مربوط به فاز دوم پروژه، لازم است تا در مورد تغییرات اعمال شده در فاز اول جهت بهبود مدل سنسور و حرکت توضیحاتی را ارئه دهیم. در فاز اول پروژه در قسمت مدل سنسور، تابعی که در نظر گرفته شده به صورت $d = \alpha z^{\beta} + \gamma$ بود که در آن d فاصله تا مانع و z مقدار خوانده شده توسط سنسور است.

ولی در فاز دوم پروژه با استفاده از این مدل سنسور جواب قابل قبولی به دست نیامد. بنابراین مجبور به تغییر تابع فوق شدیم. تابعی که در حالت جدید در نظر گرفته شده به صورت پاره‌ای خطی بود. بدین صورت که مثلاً بین دو نقطه‌ی صفر و یک سانتی‌متر یک خط عبور دادیم و به همین ترتیب بین نقاط ۱و۲ سانتی‌متر و این کار را ادامه دادیم تا نقاط ۴و۵ سانتی‌متر. به این ترتیب توانستیم با خواندن مقدار خاصی از سنسور، فاصله‌ی متناظر ربات از مانع را بدست بیاوریم. شکل (۱) مدل سنسور به دست آمده برای سنسور اول را نشان می‌دهد.

در قسمت مدل حرکتی نیز برای هریک از خطاهای جابجایی در جابجایی، چرخش در جابجایی، جابجایی در چرخش و چرخش در چرخش، واریانس خطا را بدست آوردیم. بر اساس الگوریتم ارائه شده در کتاب (شکل۲) جهت به دست آوردن مدل حرکتی، مقادیر انحراف معیار بدست آمده به جای $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ متناظر آن قرار داده شد. البته خطاهای جابه‌جایی در جابه‌جایی، چرخش در جابجایی و چرخش در چرخش دارای میانگین صفر نبودند که مقادیر میانگین آن ها نیز در تابع گوسی در نظر گرفته شد.



شکل۱- مدل سنسور اول ربات

Algorithm sample_motion_model_odometry(u_t, x_{t-1})**:**

$$\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$

$$\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^s}$$

$$\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$$

$$\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \textbf{sample}(\alpha_1 \delta_{\text{rot1}} + \alpha_2 \delta_{\text{trans}})$$

$$\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \textbf{sample}(\alpha_3\ \delta_{\text{trans}} + \alpha_4 (\delta_{\text{rot1}} + \delta_{\text{rot2}}))$$

$$\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \textbf{sample}(\alpha_1 \delta_{\text{rot2}} + \alpha_2 \delta_{\text{trans}})$$

$$x' = x + \hat{\delta}_{\text{trans}}\ \cos(\theta + \hat{\delta}_{\text{rot1}})$$

$$y' = y + \hat{\delta}_{\text{trans}}\ \sin(\theta + \hat{\delta}_{\text{rot1}})$$

$$\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$$

return $x_t = (x', y', \theta')^T$

شکل۱- الگوریتم مدل حرکتی ربات بر اساس ادومتری

با توجه به این که α_1 تا α_4 مقادیر بسیار کوچکی هستند مدل حرکتی بسیار دقیق بدست می‌آمد و ذرات به خوبی پخش نمی‌شدند، بنابراین مجبور شدیم خطا را بیش‌تر از آنچه که بود در نظر بگیریم. به همین منظور هریک از α_i ها را در ۳۰ ضرب کردیم تا میزانی خطا در مدل حرکتی ایجاد کنیم. مقادیر α_i ها پس از اصلاح و مقادیر میانگین خطاها در جدول زیر نشان داده شده اند.

نوع خطا	محیط واقعی		محیط شبیه سازی	
	α_i	میانگین	α_i	میانگین
چرخش در چرخش	$\alpha_1 = 0.0429$	0.0260	$\alpha_1 = 0.0429$	0
چرخش در جابجایی	$\alpha_2 = 1.47$	0.0019	$\alpha_2 = 1.47$	0
جابجایی در جابجایی	$\alpha_3 = 0.1415$	0.0143	$\alpha_3 = 0.1415$	0
جابجایی در چرخش	$\alpha_4 = 0$	0	$\alpha_4 = 0$	0

با توجه به کوچک بودن مقدار خطای جابجایی در چرخش α_4 صفر در نظر گرفته شد.

۲- اهداف فاز دوم پروژه

حال در ادامه به بیان کارهایی که در فاز دوم پروژه می‌بایست انجام دهیم می‌پردازیم و روش مورد استفاده را شرح می‌دهیم. دو مسئله‌ی مطرح شده در این پروژه به ترتیب سختی عبارتند از:

۱- مکان یابی ربات: در این حالت ربات نقشه را می داند اما نمی داند که در کجای آن نقشه قرار دارد. در نتیجه ربات مجبور است در محیط حرکت کرده و با داده هایی که از محیط دریافت می کند اطلاعات خود را با استفاده از یکی از روش‌های فیلتر بیزی به روز رسانی کند و مکان خود را بعد از طی مدت زمانی به دست آورد. به این مساله مکان یابی ربات[۱] می گویند. اگر ربات تاحدودی از مکان اولیه‌ی خود اطلاع داشته باشد در این صورت می‌توان از روش فیلتر کالمن و یا فیلتر کالمن توسعه یافته استفاده کرد. زیرا شرط استفاده از این روش‌ها این است که از میانگین و واریانس اولیه‌ی متغیرهای حالت ربات اطلاع داشته باشیم. به عبارتی این روش‌ها برای دنبال کردن موقعیت ربات می‌توان مناسب باشد. البته در صورتی که برای مکان اولیه‌ی ربات فرض‌های متفاوتی

[۱] Global Localization

داشته باشیم در این حالت می‌توان از فیلترکالمن چند فرضی نیز استفاده کرد. ولی اگر هیچ اعتقاد اولیه‌ای از مکان ربات نداشته باشیم یعنی دانش اولیه ربات نسبت به مکان خود یک توزیع یکنواخت باشد در این صورت استفاده از فیلتر ذرات پیشنهاد می‌شود که ما نیز در این پروژه از همین روش استفاده نمودیم.

۲- دزدیده شدن ربات: فرض کنیم ربات در محیط حرکت کرده و با استفاده از استفاده از یکی از روش های فیلتربیزی، مکان خود را پیدا کند. حال اگر ربات از آن محل برداشته شود و به محل دیگری انتقال یابد که دانش جمع شده آن با مقادیری که از حسگر های خود در حالت جدید می‌خواند فاصله زیادی داشته باشد (دارای واریانس زیادی باشد) به این اتفاق دزدیده شدن ربات[۲] گفته می شود. این اتفاق خاص می تواند خاموش کردن ربات و سنسورهایش توسط موجود دیگری مانند انسان و انتقال آن به مکان دیگری باشد.

۳- بررسی کاربردها، مزایا و معایب روش‌های فیلتر بیزی

در این قسمت به بررسی مزایا و معایب هر کدام از روش های فیلتر بیزی و کاربرد آن‌ها بر روی هر مساله اشاره خواهیم کرد. همانطور که قبلاً نیز اشاره کردیم EKF، UKF در صورت گوسی بودن توزیع ورودی و خروجی جواب خوبی به دست می‌دهند. این روش‌ها می توانند برای مسائل تعقیب موقعیت ربات به کار گرفته شوند ولی اگر میزان غیر خطی بودن نگاشت توزیع ورودی به خروجی زیاد باشد، دقت مناسبی نخواهند داشت. اگر در چند مکان از نقشه احتمال حضور ربات بالا باشد در این صورت می‌توان از این روش‌ها استفاده نمود که به آن روش فیلتر کالمن چند فرضی گفته می‌شود. با توجه به این که ما در این پروژه هیچ دانشی از مکان اولیه ربات نداریم استفاده از این روش‌ها برای ما جوابگو نخواهد بود. روش دیگری که برای حل مساله‌ی مکان‌یابی عمومی توصیه می‌شود روش فیلتر ذرات است. در این پروژه برای مسائل طرح شده از روش فیلتر ذرات استفاده می‌کنیم که دلیل این انتخاب را در زیر بیان می‌کنیم:

۱- امکان نگه‌داری و به روز کردن چند فرض اولیه به صورت همزمان.

۲- سادگی پیاده‌سازی الگوریتم و سرعت مناسب آن.

۳- قابلیت حل مسئله حتی در مدل‌هایی که در آن‌ها نگاشت از ورودی به خروجی غیرخطی است.

۴- توانایی حل مسئله‌ی مکان‌یابی فراگیر و مسئله‌ی دزدیده شدن ربات.

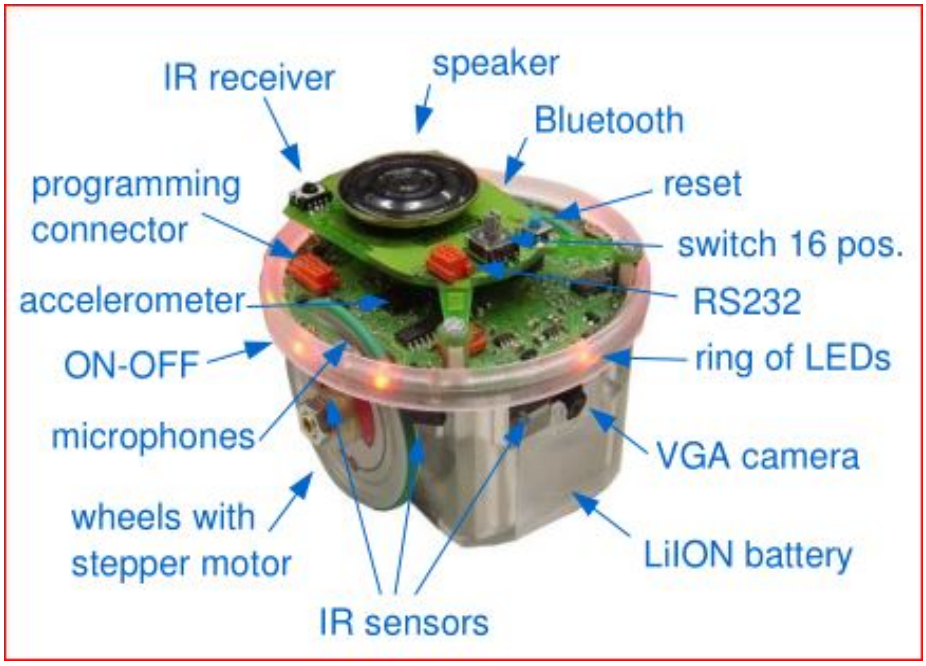
---

[۲] Kidnapping

البته این روش در صورتی جواب قابل قبولی ارائه می دهد که ابعاد حالات بالا نباشد.

۴- ربات مورد استفاده

ربات e-Puck دارای مجموعه غنی از حسگرهای ارزان قیمت است که دارای حجم کوچکی هستند. این ربات توسط دانشگاه EPFL سوییس طراحی شده است و برای کارهای تحقیقاتی مورد استفاده قرار می گیرد. این ربات دارای دو چرخ فرمان پذیر است و از طریق آنها می توان به ربات دستورهای حرکت کردن در یک محیط دو بعدی را صادر کرد. در مکانیک به این نوع کنترل، هدایت تفاضلی[۳] می گویند. حسگر های این ربات شامل ۸ حسگر Infra-red، ۳ میکروفون، یک دوربین VGA و شتاب سنج است. این ربات ابزارهای کمکی دیگری نیز دارد که حسگر نیستند. از این ابزارها می توان به اسپیکر و حلقه‌ی LED نام برد. در شکل زیر ربات e-Puck با تمام قابلیت هایش مشاهده می شود.

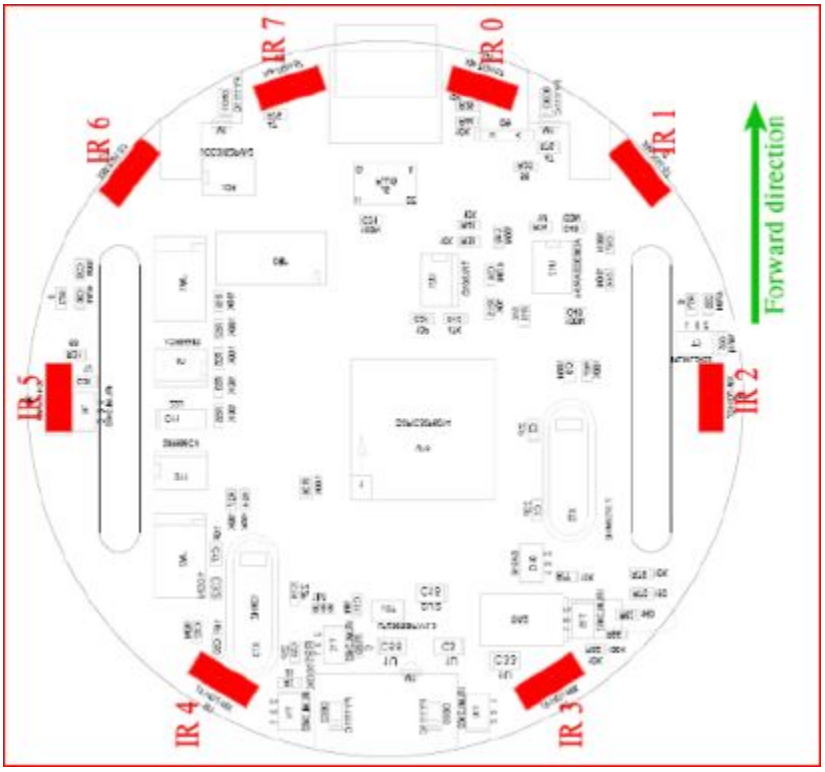


شکل۳- شکل کلی ربات

در این پروژه همانطور که در فاز اول نیز مطرح شد از حسگر دوربین وحسگر میکروفون استفاده نخواهیم کرد. بنابراین تنها از ۸ حسگر Infra-red استفاده خواهد شد. شکل زیر ساختار قرار گرفتن حسگرها را روی ربات e-Puck نشان می‌دهد.

---

[۳] Differential Wheeled

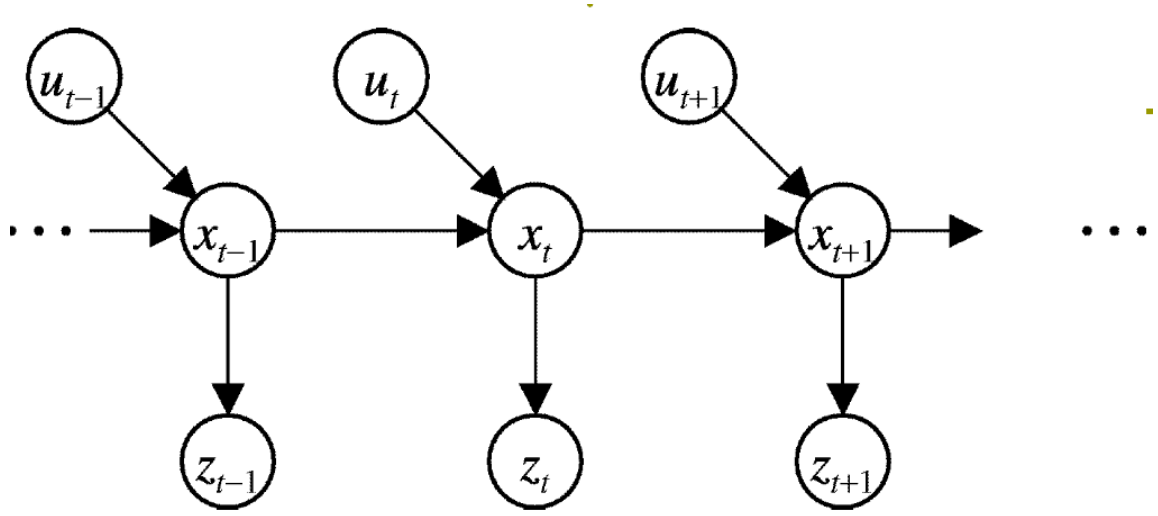


شکل۴- محل قرار گیری سنسورهای ربات

حسگرهای Infra-red ربات e-Puck دقتی در حدود ۵ سانتیمتر دارند. البته این دقت در ساعات مختلف روز تغییر می کند. در این پروژه ما مدل سنسور را طوری در نظر گرفته‌ای که اگر فاصله‌ی ربات از مانع بیشتر از ۵ سانتی‌متر شد مقدار حداکثر خوانش از سنسورها دریافت شود.

۵- مکان‌یابی ربات با استفاده از الگوریتم فیلتر ذرات

روش فیلتر ذرات یا همان روش مونت‌کارلو روشی غیر پارامتری برای تقریب مدل می‌باشد که بر اساس مدل بیزی، احتمال هر متغیر را محاسبه می‌کند. در مدل بیزی فرض می‌شود که متغیرهای مسئله براساس فرض مارکوف با یکدیگر در ارتباطاند. در مسئله‌ای که ما در این پروژه با آن مواجهیم متغیرهای مسئله X (مکان ربات)، u (ورودی کنترلی)، Z (مشاهدات ربات) به صورت زیر با هم در تعامل‌اند.



شکل ۵- نحوه ارتباط u، x و z

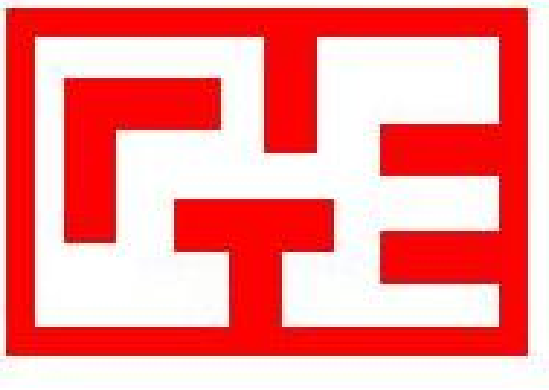
یعنی در هر گام ابتدا فاز پیشبینیT یعنی حرکت ربات متناسب با مدل حرکتی ربات انجام می‌شود و سپس با استفاده از اطلاعاتی که از سنسورها به دست می‌آید فاز تصحیح را انجام می‌پذیرد.

در الگوریتم فیلتر ذرات نیز ابتدا ربات یک اندازه گیری با استفاده از حسگرهای خود می‌کند. سپس یک عمل را انجام می دهد و مقدار odometry را به عنوان ورودی بر روی ذرات تصادفی تولید شده اعمال می‌کند. پس از حرکت ذرات درستی نمایی مقادیر حس شده توسط حسگر های ربات با مقدار حس شده توسط ذرات با استفاده از الگوریتم پرتو نگاری محاسبه می شود. جزئیات الگوریتم پرتونگاری در قسمت بعدی توضیح داده خواهد شد. الگوریتم پیاده‌سازی شده در پروژه به طور کلی به صورت زیر است:

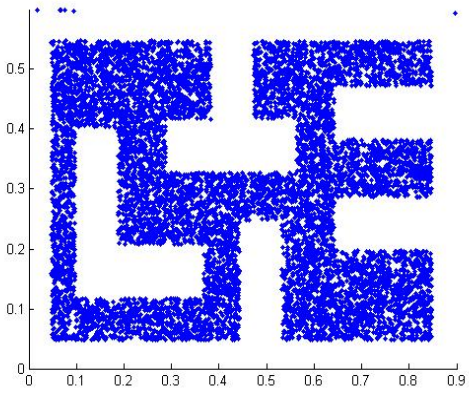
۵-۱- ایجاد ذرات اولیه به صورت تصادفی:

در گام اول تعدادی ذره را به طور یکنواخت در کل فضای نقشه پخش می‌کنیم که در آن برای هر ذره یک موقعیت و یک جهت اولیه در نظر می‌گیریم. در هنگام ساخته شدن ذرات باید به این نکته دقت کنیم که ذره‌ی ساخته شده نباید با دیواره‌ی موجود در محیط تداخل داشته باشد. همچنین در راستای پخش بهینه ذرات اولیه موانع موجود در محیط به اندازه شعاع ربات بزرگتر می شوند تا از تولید ذرات در نقاط غیر قابل استفاده جلوگیری شود.

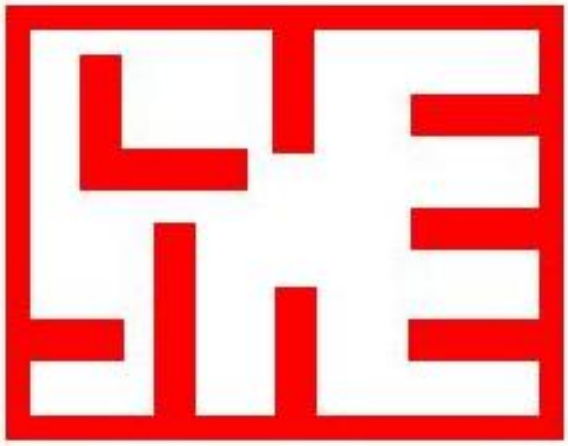
شکل ۶ و ۷ به ترتیب یک نمونه از محیط با موانع بزرگ شده و مکان اولیه ذرات را وقتی که اندازه‌ی موانع بزرگ‌تر شده‌اند، را نشان می‌دهد. همچنین شکل‌های ۸ و ۹ موارد فوق را در محیط آزمون نشان می‌دهد.



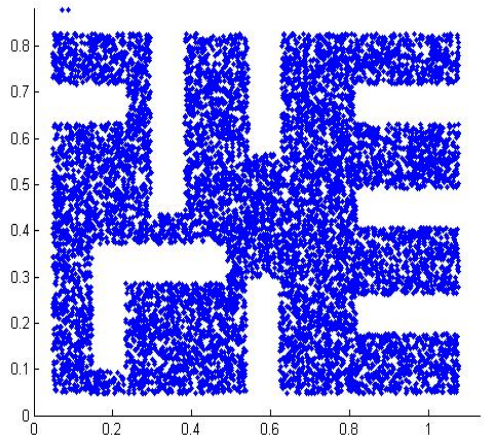
شکل ۶



شکل ۷



شکل ۸



شکل ۹

فرایند بررسی تداخل باعث می‌شود که تنها ذرات معتبر ساخته شوند و ذرات غیر مفید تولید نشوند.

۵-۲- حرکت ربات و ایجاد ذرات جدید

در این مرحله که فاز پیش‌بینی نامیده می‌شود بعد از حرکت ربات در محیط، هر ذره نیز متناسب با مقدار خوانده شده از ادومتری و با توجه به مدل حرکتی در نظر گرفته شده برای ربات در محیط شروع به حرکت می‌کند. با توجه به این که در حرکت نویز وجود دارد ذرات به یک میزان حرکت نمی‌کنند.

۵-۳- به دست آوردن وزن ذرات و نمونه برداری مجدد از ذرات

در این مرحله می‌بایست وزن هر ذره قبل از فرایند نمونه‌برداری مجدد به دست آورد. برای بدست آوردن وزن‌ها باید ابتدا مقادیر سنسورها را از ربات‌ها خواند و سپس فرایند پرتو نگاری[۴] را برای هر ذره انجام داد. با مقایسه نتایج پرتو نگاری با مقدار واقعی ربات و بسته به مقدار شباهت، یک وزن را با توجه به مدل سنسوری به آن ذره اختصاص داده می‌شود. نحوه وزن دهی به ذرات بر اساس الگوریتم موجود در کتاب (شکل ۱۰) است. که در آن

$$P_{hit}\left(z_t^k \mid x_t, m\right) = \frac{1}{\sqrt{2\pi 1.5^2}} e^{-\frac{\left(z_t^k - z_t^{k*}\right)^2}{2\times 1.5^2}}$$

$$P_{\max} = 0.8$$

$$P_{rand} = 0.2$$

و ضرایب Z_{hit} ، Z_{max} و Z_{rand} به صورت زیر انتخاب شده‌اند:

نوع مشاهده سنسور	Z_{hit}	Z_{max}	Z_{rand}
سنسور بیشترین فاصله را نخواند	7.5	0	0.2
سنسور بیشترین فاصله را بخواند	0	0.9	0.2

[۴] Ray casting

1: **Algorithm beam_range_finder_model**(z_t, x_t, m):

2: $q = 1$

3: *for* $k = 1$ *to* K *do*

4: *compute* z_t^{k*} *for the measurement* z_t^k *using ray casting*

5: $p = z_{\text{hit}} \cdot p_{\text{hit}}(z_t^k \mid x_t, m) + z_{\text{short}} \cdot p_{\text{short}}(z_t^k \mid x_t, m)$

6: $+ z_{\text{max}} \cdot p_{\text{max}}(z_t^k \mid x_t, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k \mid x_t, m)$

7: $q = q \cdot p$

8: *return* q

شکل ۱۰- الگوریتم به دست آوردن وزن ذرات

مدل سنسوری را در فاز قبلی پروژه توضیح دادیم و حال می‌خواهیم فرایند پرتو نگاری را توضیح دهیم:

۵-۳-۱- پرتو نگاری

در پرتو نگاری باید اطلاعات فاصله سنسور های نصب شده روی ربات تا موانع روبروی آنها را استخراج کنیم. بدین منظور از اطلاعات نقشه استفاده خواهیم کرد. روند کلی الگوریتم به این صورت است که از موقعیت سنسورها و در راستای شعاع مربوط به هر سنسور که از جمع زاویه سنسور نسبت به جلوی ربات و زاویه ربات در فضا به دست می آید به صورت گام به گام جلو می رویم تا زمانی که به مانع یا ماکزیمم فاصله قابل قبول برسیم و سپس فاصله تا مانع بر حسب سانتی‌متر به خروجی داده خواهد شد.

زاویه بین سنسور ها و جلوی ربات با استفاده از شکل ربات استخراج شده و برای سنسور های ۰ تا ۷ برابر با مقادیر زیر است:

RobotSensorsAngle={5.9614,5.4340,4.7124,3.6052,2.6779,1.5708,0.8491,0.3217};

گام تغییرات در راستای x برابر با کسینوس زاویه سنسورها در دستگاه مختصات کلی و در راستای y برابر با سینوس این زاویه است. بعد از اضافه کردن این گام، رد یک متغیر جانبی دیگر مقدار گرد شده مقادیر را

محاسبه می کنیم و در صورتی که نقطه بدست آمده در نقشه بیانگر مانع باشد فاصله نقطه تا سنسور را به عنوان خروجی تابع بر می گردانیم. چنانچه فاصله نقطه جدید تا موقعیت سنورها بیشتر از مقدار قابل قبول برای سنسورها باشد مقدار بیشنه قابل قبول برای سنسورها را به عنوان خروجی منظور خواهد شد. این عمل برای هر کدام از سنسورها به صورت مجزا انجام می‌شود و نهایتا یک آرایه ۸ عنصری به عنوان فاصله سنسورها تا مانع یا به عبارتی مقدار صحیحی که هر سنسور باید برگرداند محاسبه می‌شود.

بعد از این که وزن هر ذرات به دست آمد، به طور تجربی مشاهده گردید که اضافه نمودن یک مقدار ثابت به وزن ها نتیجه بهتری در بر خواهد داشت وذرات با وزن کم سریع از بین نخواهند رفت. این مقدار ثابت در هر مرحله برابر با بیشینه وزن ذرات در نظر گرفته شد. سپس این وزن‌ها را نرمالیزه می‌کنیم و با استفاده از الگوریتم موجود در کتاب (شکل ۱۱) ذرات نمونه برداری می‌شوند. مزیت این الگوریتم عدم از بین بردن سریع ذرات با وزن کم است.

Algorithm Low_variance_sampler($\mathcal{X}_t, \mathcal{W}_t$):
$\bar{\mathcal{X}}_t = \emptyset$
$r = \text{rand}(0; M^{-1})$
$c = w_t^{[1]}$
$i = 1$
for $m = 1$ *to* M *do*
  $U = r + (m-1) \cdot M^{-1}$
  while $U > c$
    $i = i + 1$
    $c = c + w_t^{[i]}$
  endwhile
  add $x_t^{[i]}$ *to* $\bar{\mathcal{X}}_t$
endfor
return $\bar{\mathcal{X}}_t$

شکل ۱۱- الگوریتم نمونه برداری

۶- مسئله ربوده شدن ربات (Kidnapping)

مسائلی که در قسمت فوق مطرح شد چارچوب کلی روش فیلتر ذرات برای مکان‌یابی عمومی بود. حال می‌خواهیم الگوریتمی که برای حل مسئله‌ی دزدیده شدن ربات مورد استفاده گردید را به طور خلاصه توضیح دهیم.

برای حل مسئله‌ی دزدیده شدن ربات از الگوریتم موجود در کتاب (شکل ۱۰) استفاده شد. این الگوریتم کمک می‌کند تا دزدیده شدن ربات تشخیص داده شود و در نتیجه ذرات را مجدداً با توزیع یکنواخت در فضا پخش کنیم و دوباره از همان الگوریتم مکان‌یابی عمومی برای یافتن مکان واقعی ربات استفاده کنیم.

1: **Algorithm Augmented_MCL**($\mathcal{X}_{t-1}, u_t, z_t, m$):
2: static $w_{\text{slow}},\ w_{\text{fast}}$
3: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
4: for $m = 1$ to M do
5: $x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
6: $w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$
7: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
8: $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M}\ w_t^{[m]}$
9: endfor
10: $w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$
11: $w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$
12: for $m = 1$ to M do
13: with probability $\max(0.0,\ 1.0 - w_{\text{fast}}/w_{\text{slow}})$ do
14: add random pose to $\mathcal{X}_t$
15: else
16: draw $i \in \{1, \ldots, N\}$ with probability $\propto w_t^{[i]}$
17: add $x_t^{[i]}$ to $\mathcal{X}_t$
18: endwith
19: endfor
20: return $\mathcal{X}_t$

شکل فوق الگوریتم تطبیقی مورد استفاده‌ی ما در حل مسئله‌ی دزدیده شدن ربات بود که در آن وزن‌های آهسته و سریع برای هر مرحله محاسبه می‌شود. زمانی که $P_{rand} = 1 - \frac{w_{fast}}{w_{slow}} > 0$ به دست آمد، در کل فضا ذرات را از نو به طور یکنواخت پخش می‌کنیم. از طرفی ما حد فوق را برای P_{rand} صفر قرار ندادیم و مقدار آن را با سعی و خطا بدست آوردیم که ممکن است از یک محیط به محیط دیگر فرق کند. برای محیطی که ما آزمایش‌هایمان را روی آن انجام دادیم مقدار 0.8 دست آمد و در محیط تحویل پروژه نیز مقدار آن را 0.75 قرار

دادیم. بخش دیگری از حل این مسئله پیدا کردن α_{fast} و α_{slow} بود که مقادیر آن را نیز با سعی و خطا بدست آوردیم. از طرفی مقادیر آن‌ها طوری انتخاب شده بود تا نسبت آنها برابر 10 شود.

نتایج

در این پروژه قصد داشتیم تا مسئله‌ی مکان‌یابی عمومی را برای یک ربات epuck در یک محیط واقعی حل کنیم. سیستم سنسوری و حرکتی که از آن‌ها استفاده کردیم به ترتیب شامل هشت سنسور IR و دو چرخ بود که توسط هدایت تفاضلی کنترل می‌شوند و هر کدام دارای یک انکدر می‌باشند. در فاز اول پروژه هدف پروژه هدف تعیین مدل سنسوری و حرکتی ربات بود که با انجام شبیه‌سازی‌های مختلف توانستیم با دقت خوبی آن‌ها را بدست بیاوریم.

در فاز دوم پروژه مسئله‌ی مکان‌یابی را با استفاده از مدل‌های بدست آمده از فاز یک حل کردیم. از طرفی مسئله‌ی دزدیده شدن ربات را که حکم امتیازی در این پروژه داشت حل کردیم و توانستیم مکان ربات را پس از دزدیده شدن دوباره پیدا کنیم. در هر دو مسئله‌ی فوق از روش فیلتر ذرات استفاده نمودیم.(به دلیل نیاز به داشتن تعداد بسیار زیادی فرض اولیه، غیر خطی بودن توابع تبدیل، توانایی حل مسئله‌ی دزدیده شدن ربات و سرعت و سادگی الگوریتم)

مراجع

[1] https://www.udacity.com/course/cs373 by S.Thrun

[2] S.Thrun, W.Burgard, D.Fox, Probabilistic Robotics 2000

[3] Webots User Guide release 6.4.3