Guião SAut

Pioneer 3-DX with EKF-SLAM

[Slide 2] The Project:

- O objetivo do nosso projeto foi implementar um algoritmo de SLAM, simultaneamente localização e mapeamento com um robot Pioneer 3-DX.
- Para isto utilizou-se Python para desenvolver o código necessário, beneficiando da sua sintaxe e de todas as bibliotecas disponíveis open source.
- Como podemos ver, o Pioneer 3-DX tem 2 rodas motorizadas e uma roda auxiliar com mobilidade total o que lhe permite deslocar-se em frente e/ou rodar sobre si próprio.

[Slide 3,4] Sensors:

- Tem odómetros em ambas as rodas o que lhe permite prever a distância percorrida num intervalo de tempo, por cada roda e assim fazer uma estimativa da sua posição atual. Tem também 8 sonares no painel dianteiro e mais 8 no traseiro, que podem ser usados para detetar objetos, paredes em seu redor.
- O nosso grupo, no entanto, decidiu utilizar um laser Hokuyo, conectado por USB e colocado na parte superior dianteira do robot, para realizar estas medições do ambiente em redor.

[Slide 5] Hokuyo Laser:

- Este Hokuyo Laser é alimentado pela mesma ligação USB, Tem uma precisão de 3 centímetros para medições entre 6 cm a 1 metro, e de 1 a 4 metros uma incerteza de 3% (chegando a 12 cm de incerteza para 4 metros). Cada medição demora 100 milissegundos.

[Slide 6] Microsimulation:

- Para testar o nosso algoritmo utilizamos vastamente o Gazebo que nos permitiu simular a performance do nosso robot através do ROS de uma maneira muito próxima da real. Para isto utilizamos um modelo do Pioneer para Gazebo de um repositório do Mario Serna.

[Slide 7] The algorithm:

- O algoritmo de SLAM implementado foi o Extended Kalman Filter. O pseudocódigo deste algoritmo encontra-se neste slide e inclui duas etapas principais: a etapa de predição e a etapa de correção. Durante a primeira, o robot move-se e a nova estimativa do predicted state é estimada a partir da anterior; durante a segunda etapa, o laser observa as landmarks já conhecidas de forma a corrigir o mapeamento feito.

[Slide 8] Odometry & prediction step:

- O modelo da odometria assume que o movimento do robot que ocorre entre dois instantes de tempo consecutivos pode ser decomposto numa primeira rotação, seguida de uma translação e de uma segunda rotação.
- De forma a fazer uma simulação o mais parecida possível com o mundo real, procedeu-se à introdução de ruído.
- O modelo usado assume a adição de ruído à translação e às rotações
- Há 4 parâmetros distintos que quantificam a influência que o ruído de cada um destes movimentos tem nele próprio e nos outros dois

[Slide 9] Landmarks:(E)

Quanto ao tipo de Landmarks, usamos no nosso algoritmo tanto linhas como cantos. As linhas têm a vantagem de serem fáceis de identificar, usando métodos como aquele que vamos apresentar de seguida, dão uma ideia mais ampla do que rodeia o robot. Bastante apropriadas para o ambiente onde o robot estará. Isto pode ser visto como uma desvantagem, visto que tentamos sempre ao máximo não limitar o nosso algoritmo a um ambiente específico. Os cantos têm a vantagem de minimizar o número de parâmetros que os descrevem, o que têm um grande impacto na complexidade do cálculo do EKF. São mais sujeitos a ruído e requerem que os pontos se liguem uns aos outros.
Começamos portanto por usar linhas no processo de interpretação dos dados do laser.

[slide 10] Landmarks detection & Hough transform:(E)

- Utilizamos o Hough transform para identificar e isolar certos recursos importantes representados numa imagem, neste caso linhas.

- Os pontos no espaço de Hough estão na forma P(ρ,ɵ), em que a coordenada-x desse ponto é dado pela distância (ρ) da linha do r a origem, e a coordenada-y corresponde ao ângulo de inclinação (ɵ) da linha normal para r que passa pela origem. Isto é, cada ponto no espaço de coordenadas corresponde a uma reta no Hough Space e é através das interseções de retas que se detetam retas.

[slide 11] Hough Transform(E)

- Foram testados dois métodos distintos da biblioteca “OpenCV” para implementar esse algoritmo.
- É necessário escolher certos parâmetros relativos à sensibilidade do método.
- O threshold é um deles e corresponde ao número mínimo de interseção no espaço dos parâmetros para considerar uma linha.
- Decidimos por utilizar o método probabilístico, porque os resultados obtidos foram mais precisos, e os pontos de partida e final de cada linha são dadas directamente, e eles serão assumidos como cantos
-

[Slide 12] Synchronization problem:(P)

- Depois de implementarmos o algoritmo, começamos a fazer alguns testes e reparamos que as landmarks que tínhamos não tinham nada a ver com as que estavam no mundo que construímos.
- Durante vários dias pensamos que pudesse ser o algoritmo que estivesse mal implementado e portanto revimos linha a linha o nosso código várias vezes mas não conseguimos resolver o problema. O passo seguinte foi suspeitar dos valores que estávamos a dar às matrizes Qt, Rt, o threshold da distância Mahalanobis e as covariâncias com que inicializamos uma matriz nova. Experimentamos tudo o que conseguimos lembrar: mudar os valores numericamente, inserir dependências na velocidade linear e angular. Mas mesmo assim, essas alterações não corrigiam o nosso state vector.

- Por isso, já quase no desespero decidimos tentar imaginar todo o tipo de problemas que pudessem estar a afetar o nosso código.

No final chegamos a 2.

- Um problema que demorou bastante tempo até ser detectado foi o problema da sincronização.

- Este problema aconteceu porque os dados do laser e da odometria com que o update processava correspondiam a timestamps diferentes. Como se pode ver pelo esquema, os dados do laser demoravam mais tempo a ser processados pelo Hough transform, do que os dados da odometria demoravam na prediction calculation. Isso fazia com que o nosso algoritmo calculasse o state vector com observações antigas e uma pose atual. Isso num movimento tão simples com uma simples rotação no sentido contra-relogio faz com que as landmarks estejam tenham um desfasamento cumulativo no sentido contra relógio também.

Explicar imagem:

[Slide 13] State machine implementation(P)

- Para resolver esse problema experimentamos fazer uma espécie de prediction no update para estimar a posição do robô na altura dos lasers, mas isso não resultou e por isso tivemos de implementar uma máquina de estados com o esquema que a figura mostra para termos a certeza que não existe nenhum desfasamento cumulativo e seja apenas um desfasamento constante de valor praticamente insignificante(0.1 + processamento da odometria).

Explicar esquema

[Slide 14] Discontinuity problem:(P)

O 1º é o Discontinuity Problem:

- No update measurement, quando o algoritmo calcula o ângulo que o laser devia medir para uma landmark m que está em coordenadas cartesianas, ele usa a função atan2 que é descontínua em pi e -pi.
- Assim, como podem observar pelas imagens, durante o matching step, quando o algoritmo está a medir o delta z que é a diferença entre o que mede e o que era suposto medir, se algum desses Zs estiver numa vizinhança da descontinuidade existe a possibilidade do vetor delta Z está a ser calculado da maneira errada. Como acontece neste exemplo. O vetor z dessa landmark está numa zona

descontínua o que faz com que o delta z não seja nulo e por conseguinte o algoritmo classifica essa landmark como nova.

[Slide 15-18] Microsimulation results:(M)

- Apresentação dos resultados obtidos no Gazebo e identificação de novas landmarks

[Slide 19] Other results:(M)

- Neste slide apresentam-se os resultados obtidos diretamente da prediction step do algoritmo EKF. Na figura do lado esquerdo apresenta-se a comparação da predição da pose do robot quando o modelo contém ruído com a pose real do robot. Não seria de esperar esta diferença significativa.
- Na figura da direita, a elipse corresponde a uma representação da matriz de covariância: os eixos da elipse estão relacionados com os valores próprios da matriz e a orientação da elipse com os vetores próprios. A elipse apresenta um alongamento segundo a direção em que o robot se está a mover

[Slide 20] Experimental results :(M)

- Esta imagem contém a planta real do corredor, na qual se sobrepuseram as features detetadas a partir do *rosbag* efetuado. O ponto vermelho corresponde à posição do robot e os pontos azuis à posição das landmarks

“” Para vender
Assim, cansados e quase sem esperança que fossemos conseguir corrigir a tempo os problemas decidimos acreditar que o algoritmo estava bem implementado e tentamos procurar outras hipóteses que pudessem justificar os nossos problemas. E no final, após conseguirmos imaginar, construir e testar um cenário que permitisse avaliar cada hipótese individualmente sem nós, conseguimos isolar, identificar e resolver os problemas.
“”