Technical Report
Reference Frames, Sensing, and Asynchronous Mapping with Coppelia Simulator

Alysson Ribeiro da Silva
Computer Science Graduate Program
Federal University of Minas Gerais
Belo Horizonte, Brazil

Abstract—Desastres naturais, como tsunamis e enchentes, são fonte recorrente de danos sociais e econômicos ao redor do mundo. Com isso em mente, este documento técnico apresenta a proposta de uma API para controle, comunicação, mapeamento, e sensoriamento, utilizando robôs móveis simulados através do CoppeliaSim. Quatro testes foram feitos para validar a API em termos de capacidade de lidar com os dados provenientes do simulador para navegação através de um algoritmo simples de detecção de obstáculos. Os resultados obtidos indicam que a API possui robustez e permite a comunicação consistente com o simulador.

Index Terms—Coppelia Simulator, Reference Frames, Mapping,

1 Introdução

Desastres naturais, como tsunamis e enchentes, são fonte recorrente de danos sociais e econômicos ao redor do mundo, por consequência da alta taxa de mortalidade [2], [4], custos com reconstrução, e reparos na infraestrutura das regiões afetadas. Como exemplo, os tsunamis e enchentes [7] estão dentre os tipos mais devastadores de desastres, conseguem cobrir grandes quantidades de terras em curtos períodos de tempo e causar danos substanciais [3]. Diferentes abordagens vem sendo tomadas para tentar mitigar as consequências dos desastres naturais [1], o que inclui o uso de robôs móveis capazes de navegar, sentir, e mapear ambientes perigosos em missões de busca e resgate [5], [6].

Várias técnicas podem ser utilizar para efetuar mapeamento de ambientes e também para controle de robôs móveis terrestres em um ambiente sem variações de altitude. Uma vez que utilizar robôs reais nem sempre é possível, o objetivo deste projeto é avaliar a capacidade do simulador CoppeliaSim para controle, sensoriamento, e mapeamento em tempo real de um ambiente simulado. Para se comunicar com o simulador, permitindo sua avaliação, foi desenvolvida uma API que permite gerenciar e controlar robôs móveis, adquirir, armazenar, e apresentar informações em tempo real sobre obstáculos e caminho percorrido pelos robôs, e também para permitir a construção e visualização - em tempo real - de um mapa 2D discretizado que pode ser utilizado por algoritmos de construção de caminhos.

Quatro testes foram efetuados para avaliar a API proposta. O primeiro teste visa verificar a capacidade da mesma em em armazenar, operar, e exibir os diferentes sistemas de coordenadas presentes em uma cena. Já o segundo teste vista avaliar a capacidade da mesma em obter os dados do simulador em tempo de execução e exibir os mesmos. O terceiro teste visa avaliar a capacidade de apresentar e construir uma representação do ambiente em tempo real. Por fim, o último teste avalia a porcentagem do mapa explorado e construído através de uma métrica de índice de exploração. No apêndice A são apresentadas instrução para execução dos códigos fornecidos.

2 Método

O método divide-se em três camadas, como ilustrado pela Figura 2. A primeira camada - representada pela cor laranja - é responsável por permitir manipular, visualizar os diferentes sistemas de coordenadas, visualização dos obstáculos detectados e percurso percorrido pelo robô, e montar e visualizar mapa 2D para utilização em algoritmos em tempo real. Já a segunda camada - representada pela cor roxa - permite a manipulação de um robô móvel configurado dentro do CoppeliaSim. Por fim, a última camada - representada pela cor vermelha - é composta de objetos e funções que auxiliem no tratamento dos dados obtidos do simulador.

A comunicação entre os objetos da API proposta é representado pelo diagrama presente na Figura 2, onde o objeto do tipo Robot deve ser instanciado para permitir acesso ao simulador.

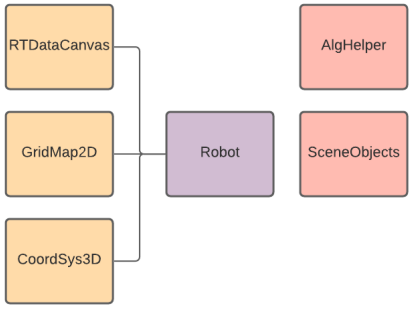

Figura 1: Objetos criados para representar o robô (em roxo), mapas e gráficos interativos (em laranjado), e operações auxiliares (em vermelho).

Uma vez instanciado, o mesmo tem a capacidade de interagir com o mundo, obter e operar diferentes sistemas de coordenadas através do CoordSys3D, apresentar dados em tempo real obtidos dos sensores e trajetória através do objeto RTDataCanvas. Além disso, um mapa de duas dimensões pode ser construído através do objeto GridMap2D e medições do sensor a laser.

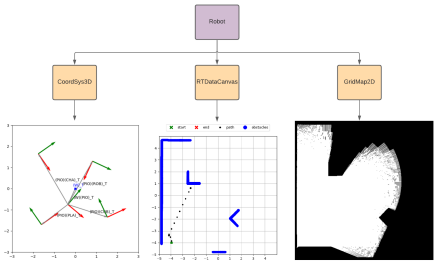

Figura 2: Responsabilidades dos objetos criados para auxiliar a manipulação e reconhecimento do ambiente.

2.1 Representando e Configurando um Robô

Nesta proposta um robô móvel terrestre é representado pelo objeto Robo. O mesmo é responsável por armazenar cada `handle` proveniente do CoppeliaSim, assim como variáveis de controle de sua simulação, comportamentos para navegação, objetos para visualização e mapeamento. Além disso, o mesmo também possui várias funções que possibilitam a manipulação de seus atuadores e interação com os outros objetos do sistema.

2.2 Representando Matrizes Homogêneas

As matrizes de transformação homogêneas - utilizadas para manipular e extrair informações sobre o ambiente - são criadas através do objeto CoordSys3D no seguinte formato,

$$ {}^{X}_{O}T = \begin{bmatrix} cos(\theta)\zeta & -cos(\theta) & 0 & {}^{X}P_x \\ sin(\theta) & cos(\theta)\zeta & 0 & {}^{X}P_y \\ 0 & 0 & 1 & {}^{X}P_z \\ 0 & 0 & 0 & {}^{X}P_w \end{bmatrix} \quad (1) $$

onde, θ representa o ângulo de rotação do referencial do corpo rígido O em relação a um referencial X ao redor de $\hat{Z}$, XP representa o ponto de origem do referencial com relação a X, e ζ representa um fator de escala utilizado para projeção de coordenadas normalizadas em outros planos. Além da representação de matrizes, o CoordSys3D também permite calcular matrizes de transformação inversas e apresentar os sistemas de referencias e relacionamentos em um gráfico.

2.3 Apresentação de Trajetórias em Tempo Real

O objeto RTDataCanvas é proposto para apresentar os dados capturados pelos sensores do robô e também sua trajetórias no referencial global. O mesmo permite desenhar de forma interativa utilizando a biblioteca pyplot de forma a não bloquear a execução do sistema. Os gráficos são atualizados através da função `blit` que utiliza o conceito de `double buffer` para rasterização eficiente. Em primeiro momento, o RTDataCanvas é utilizado neste projeto para localização de obstáculos e cálculo de trajetória através da matriz de transformação homogênea,

$$ {}^{W}_{L}T = {}^{W}_{X}T \; {}^{X}_{L}T \quad (2) $$

onde, $\{W\}$ é o referencial global, $\{X\}$ é o referencial do robô, e $\{L\}$ representa o referencial do dado obtido do robô. Um exemplo de utilização de ${}^{W}_{L}T$, é apresentado no Algoritmo 1. O mesmo é utilizado para calcular a posição de cada feixe do sensor a laser com relação ao mundo e também para atualizar o RTDataCanvas. É importante notar que o operador @ representa a multiplicação de matrizes.

```
para cada FEIXE em LASER
    distancia = FEIXE.dist
    angulo    = FEIXE.ang

    se distancia < distancia maxima
        x = distancia * cos(angulo)
        x = distancia * sin(angulo)

        P_local = (x,y)
        P = w_l_T @ P_local

        RTDataCanvas.update(P)
```

Algoritmo 1 - Peseudo código para mapeamento dos dados do laser em tempo real.

2.4 Criando Mapas Bidimensionais em Tempo Real

Um mapa bidimensional é construído através das medições a laser provenientes do robô. Para que isso seja possível é criada uma transformação de ViewPort ${}_{VIEW}T$ com auxílio das seguintes matrizes,

$$ {}_{SCALE}T = \begin{bmatrix} \frac{1}{\beta} & 0 & 0 & 0 \\ 0 & \frac{1}{\beta} & 0 & 0 \\ 0 & 0 & \frac{1}{\beta} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3) $$

$$ {}_{DESLOC}T = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4) $$

$$ {}_{ROT}T = \begin{bmatrix} cos(90) & -cos(90) & 0 & 0 \\ sin(90) & cos(90) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5) $$

$$ {}_{MAP}T = \begin{bmatrix} width & 0 & 0 & 0 \\ 0 & height & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6) $$

onde,

$$ \begin{aligned} {}_{NORM}T &= {}_{SCALE}T\,{}_{DESLOC}T\,{}_{ROT}T \\ {}_{INTER}T &= {}_{MAP}T\,{}_{NORM}T \\ {}_{VIEW}T &= {}_{INTER}T\,{}^{W}_{X}T \end{aligned} \quad (7) $$

sendo, ${}_{NORM}T$ uma transformação de normalização do espaço entre o intervalo $[0, 1]$, ${}_{INTER}T$ representa uma transformação de mapeamento do espaço normalizado para o mapa a ser construído de tamanho $width$ e $height$, e ${}_{VIEW}T$ a transformação final que converte o espaço normalizado do mundo para o intervalo de trabalho do mapa 2D armazenado dentro do objeto GridMap2D. A intuição por trás dessas transformações é obter um ponto do espaço do simulador, normalizar o mesmo (como um vetor unitário), e projetar em uma matriz 2D de uma certa largura e altura.

Para permitir o mapeamento do laser de forma eficiente durante a simulação com o CoppeliaSim, foi utilizado o método do ponto médio. O mesmo permite construir o mapa em complexidade linear a quantidade de feixes e células entre pontos inicias e finais. É importante notar que se houver ruído nos dados do laser, caso a distância medida seja maior que a distância máxima, o algoritmo deve considerar que há um espaço em branco entre o ponto inicial e final. O resultado do sistema de mapeamento é ilustrado na Figura 3.

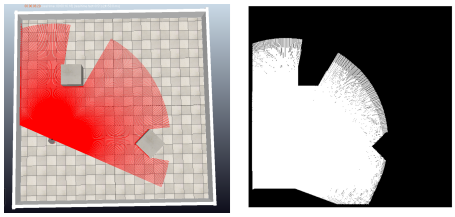

Figura 3: Sistema de mapeamento em tempo real através do GridMap2D utilizando algoritmo do ponto médio o CoppeliaSim.

3 Testes

Para avaliar o sistema proposto, 4 testes foram conduzidos. O primeiro teste vista verificar a capacidade do objeto CoordSys3D em armazenar, operar, e exibir os diferentes sistemas de coordenadas presentes em uma cena. Já o segundo teste vista avaliar a capacidade do objeto SceneObjects em obter os dados do simulador em tempo de execução e utilizar o CoordSys3D para exibir os mesmos corretamente com relação ao referêncial do robô em três posições e rotações distintas. Em seguida, o terceiro teste visa avaliar a capacidade do RTDDataCanvas e GridMap2D em apresentar e construir uma representação do ambiente em tempo real. Por fim, o último teste avalia a porcentagem do mapa explorado e construído através do GridMap2D em três execuções distintas.

3.1 Configuração Experimental

Para efetuar os testes foi utilizado o robô pioneer3dx equipado com um sensor laser Hokuyo URG 04LX. O sensor laser utiliza a API do CoppeliaSim para enviar os dados referentes a distância e ângulo de cada feixe. Para garantir o funcionamento correto do sensor em todo o contexto da simulação foi modificada a lista de distâncias construída de forma a receber a norma euclidiana com relação ao robô. Além disso, também foram removidos valores negativos que eram adicionados ao fim do comportamento do sensor que podem prejudir os algoritmos.

Para permitir a navegação foi utilizado um algoritmo simples que baseia-se na detecção de obstáculos a direita, frente, e esquerda do robô. Para detectar a distância de um obstáculo é gerado um vetor de contato $V = v_1, ...v_n$, onde $v_i \in V$ é associado a um l_i - feixe do laser - proveniente do sensor. Cada v_i recebe o valor 1 se a distância de l_i para um obstáculo for menor que um liminar e 0 caso contrário.

Com o vetor V calculado, o algoritmo simples de navegação - proposto dentro do objeto Robo - divide o mesmo em bandas, como ilustrado na Figura 4. Cada banda é associada a um lado do robô e caso algum v_i pertencente a banda for igual a 1, então ela recebe o valor 1. Esse valor

indica que pode haver colisão nas proximidades da mesma.

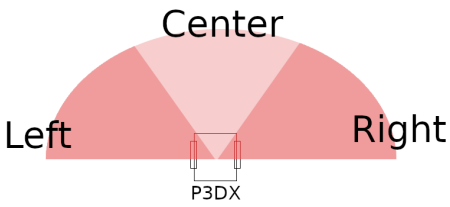

Figura 4: Representação das bandas de detecção utilizando o sensor laser presente no Robô Pioneer3dx.

O algoritmo de navegação simples consiste em girar 30 graus na direção oposta a banda com valor maior que 0. Essa proposta difere do comportamento base do robô que utiliza apenas um feixe central para detecção de possível colisão.

Para calcular o índice de exploração no terceiro e último testes é utilizada a seguinte relação,

$$i = \frac{\gamma}{\beta} \tag{8}$$

onde γ representa a quantidade de células exploradas e armazenadas em GridMap2D pelo robô e β representa o tamanho total do mapa no mundo real (simulado) após ser projetado por ${}_{VIEW}T$.

3.2 Cenas e Chamadas Assíncronas

Todas as chamadas para a API do CoppeliaSim - em tempo de execução - são feitas utilizando a opção `simx_opmode_streaming`. Essa escolha permite evitar a utilização de dados obtidos em diferentes quadros da simulação que não condizem com o estado correto do robô. Por fim, a posição e orientação dos sensores, robô, e objetos da cena são obtidos diretamente do ambiente de simulação. Essa última escolha elimina a existência de erros nas leituras.

Duas cenas foram utilizadas para os testes, sendo elas a `scene1.ttt` e `scene2.ttt`. A `scene1.ttt` foi utilizada no primeiro e segundo testes e a `scene2.ttt` nos demais. Ambas as cenas são ilustradas na Figura 5.

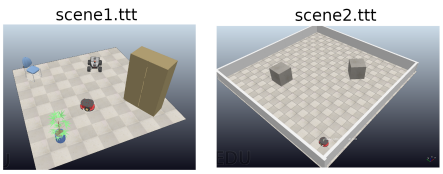

Figura 5: Representação das bandas de detecção utilizando o sensor laser presente no Robô Pioneer3dx.

3.3 Extraindo Referenciais

A cena criada para testes assim como os referênciais de cada objeto presente na mesma são apresentados na Figura 6. Onde os seguintes referenciais são representados,

$$\begin{aligned}
\{CON\} &= \{{}^{W}R_{CON}|{}^{W}P_{CON_ORIG}\} \\
\{IND\} &= \{{}^{W}R_{IND}|{}^{W}P_{IND_ORIG}\} \\
\{CUP\} &= \{{}^{W}R_{CUP}|{}^{W}P_{CUP_ORIG}\} \\
\{ROB\} &= \{{}^{W}R_{ROB}|{}^{W}P_{ROB_ORIG}\} \\
\{PIO\} &= \{{}^{W}R_{PIO}|{}^{W}P_{PIO_ORIG}\}
\end{aligned} \tag{9}$$

onde $\{CON\}$ representa o referencial da cadeira de conferências, $\{IND\}$ representa o referencial da planta, $\{CUP\}$ representa o referencial do armário, $\{ROB\}$ representa o referencial do robô Robotinik e PIO representa o referencial do Pionner3dx. Os resultados obtidos para este teste estão condizentes com a cena utilizada, logo há indícios de que as transformações e sistemas de coordenadas representados estão corretos.

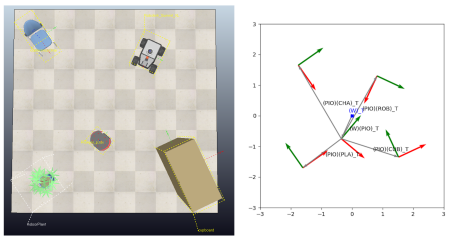

Figura 6: Apresentação da cena de testes inicial e respectivos referenciais.

3.4 Obtendo e representando referenciais em tempo de execução

Para obter as informações em tempo real é utilizada a Remote API do CoppeliaSim. Na Figura 7 são apresentadas três cenas e respectivas representações dos referenciais extraídos utilizando os objetos SceneObjects e CoordSys3D.

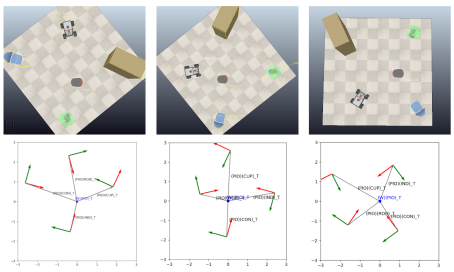

Figura 7: Captura dos referências para três posições e rotações diferentes com relação ao robô p3dx.

Todos as representações foram obtidas com relação ao referencial do robô utilizando as seguintes transformações,

$$
\begin{aligned}
{}^{PIO}_{CON}T &= {}^{W}_{PIO}T^{-1}\,{}^{W}_{CON}T \\
{}^{PIO}_{IND}T &= {}^{W}_{PIO}T^{-1}\,{}^{W}_{IND}T \\
{}^{PIO}_{CUP}T &= {}^{W}_{PIO}T^{-1}\,{}^{W}_{CUP}T \\
{}^{PIO}_{ROB}T &= {}^{W}_{PIO}T^{-1}\,{}^{W}_{ROB}T
\end{aligned}
\tag{10}
$$

onde cada transformação obtida no formato ${}^{PIO}_{X}T$ representa o referencial de X com relação ao robô. Como apresentado na Figura 7, todas as transformações estão condizentes com o posicionamento do robô e isso pode indicar o funcionamento correto dos objetos propostos.

3.5 Exploração, Apresentação, e Mapeamento Assíncrono em Tempo Real

Três execuções foram feitas para avaliar os objetos responsáveis pela apresentação e construção do mapa. Como apresentado na Figure 8,

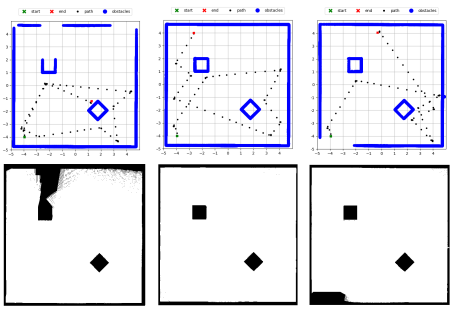

Figura 8: Execução do algoritmo de navegação simples. Em cima são apresentado os gráfico gerados pela navegação do robô e seu caminho percorrido em todas as execuções. Já na parte de baixo da figura, são apresentados os mapas de células construídos a partir do mapeamento do laser e sua projeção e escala no referencial do mapa.

onde, em cada um dos gráficos, a marca verde representa o ponto inicial, a marca vermelha representa o ponto final, o pontilhado preto representa o trajeto percorrido pelo robô, e as marcas azuis representam os obstáculos encontrados através das medições do sensor. N parte de baixo de cada gráfico é apresentado os mapas construídos para cada teste, respectivamente. No primeiro teste o robô não foi capaz de explorar completamente o mapa, logo ambas as representações estão incompletas. Porém, na segunda e terceira execuções, o robô foi capaz de explorar melhor o mapa. Uma vez que a comunicação da proposta é feita de forma completamente assíncrona, esse comportamento já era esperado, pois pequenas variações no tempo de comunicação interferem na velocidade em que o algoritmo de navegação consegue perceber mudanças no ambiente.

3.6 Índice de Exploração

A qualidade da exploração de cada teste, apresentado na Figura 8, foi medida utilizando a métrica de Índica de Exploração e seu resultado é apresentado na Figura 9. Em todos os testes o robô obteve um índice de explora acima de 80%. Esse resultado indica que aspectos como a velocidade de comunicação, apesar do tratamento dos dados de forma assíncrona, permitiu que o comportamento utilizado fosse executado de forma a não causar colisões ou travamentos. Dessa forma, o robô foi capaz de explorar os mapas de forma substancial devido ao alcance e ângulo de abertura dos sensores instalados. Além disso, esse resultado também mostra que o GridMap2D é capaz de armazenar os locais explorados de forma a permitir a utilização e aplicação de métricas e algoritmos para mensurar o desempenho dos robôs em tempo real.

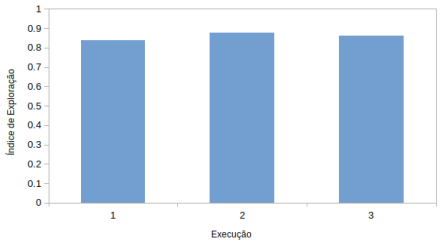

Figura 9: Índice de exploração das três execuções apresentadas no Gráfico 8.

4 Conclusão

Este documento técnico apresenta a proposta de um API para controle, comunicação, mapeamento, e sensoriamento, utilizando robôs móveis simulados através do CoppeliaSim. Foram criados vários objetos capazes de lidar com os dados obtidos dos sensores do robô de forma assíncrona que permitem visualizar o comportamento do robô em tempo real. Quatro testes foram feitos para validar a API em termos de capacidade de lidar com os dados provenientes do simulador para navegação através de um algoritmo simples de detecção de obstáculos. Os resultados obtidos indicam que a API possui robustez e permite a comunicação consistente com o simulador. Além disso, foi possível controlar o robô e gerar mapas e visualizações em tempo real sem comprometimento de performance. Como trabalhos futuros pretende-se incorporar técnicas de localização e mapeamento probabilísticas.

References

[1] Progress from catastrophe. *Nature Geoscience*, 10(8):537–537, Aug 2017.

[2] Jorge G. F. Crempien, Alejandro Urrutia, Roberto Benavente, and Rodrigo Cienfuegos. Effects of earthquake spatial slip correlation on variability of tsunami potential energy and intensities. *Scientific Reports*, 10(1):8399, May 2020.

[3] Phil R. Cummins, Ignatius R. Pranantyo, Jonathan M. Pownall, Jonathan D. Griffin, Irwan Meilano, and Siyuan Zhao. Earthquakes and tsunamis caused by low-angle normal faulting in the banda sea, indonesia. *Nature Geoscience*, 13(4):312–318, Apr 2020.

[4] Elizabeth Frankenberg, Cecep Sumantri, and Duncan Thomas. Effects of a natural disaster on mortality risks over the longer term. *Nature Sustainability*, May 2020.

[5] S. Kuswadi, S. I. Adji, R. Sigit, M. N. Tamara, and M. Nuh. Disaster swarm robot development: On going project. In *2017 International Conference on Electrical Engineering and Informatics (ICELTICs)*, pages 45–50, 2017.

[6] F. Matsuno and S. Tadokoro. Rescue robots and systems in japan. In *2004 IEEE International Conference on Robotics and Biomimetics*, pages 12–20, 2004.

[7] Dapeng Yu, Jie Yin, Robert L. Wilby, Stuart N. Lane, Jeroen C. J. H. Aerts, Ning Lin, Min Liu, Hongyong Yuan, Jianguo Chen, Christel Prudhomme, Mingfu Guan, Avinoam Baruch, Charlie W. D. Johnson, Xi Tang, Lizhong Yu, and Shiyuan Xu. Disruption of emergency response to vulnerable populations during floods. *Nature Sustainability*, May 2020.