INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Desenvolvimento de um ambiente de simulação para arquitetura robótica de um "*cable robot*"

Simão Pedro Lopes da Costa Freitas

Relatório Final da Dissertação apresentado à

Escola Superior de Tecnologia e de Gestão

Instituto Politécnico de Bragança

Para obtenção do grau de Mestre em

Engenharia Industrial

Área de Especialização de Engenharia Eletrotécnica

Orientador:

Professor Doutor José Luís Lima

Coorientador:

Professor Doutor A. Paulo Moreira, Professor Doutor Pedro G. Costa

Dezembro de 2014

FOLHA EM BRANCO

Agradecimentos

Os primeiros agradecimentos têm de ser sem qualquer dúvida para os meus pais, sem eles nada teria sido possível. O investimento, a paciência, e o apoio ao longo dos anos foram fundamentais.

Com um agradecimento especial ao meu orientador e Professor José Magalhães Lima, pelo acompanhamento e apoio no desenvolvimento deste trabalho, pela disponibilidade e sugestões para que os resultados fossem melhores, mas sobretudo por ao longo dos anos que passei nesta instituição ter sido uma referência.

Aos coorientadores Professor A. Paulo Moreira, Professor Pedro G. Costa, o agradecimento por constantemente proporem novos desafios ao longo deste projeto, o que permitiu manter a motivação para conseguir mais e melhor e ao Professor José Gonçalves pelas críticas e opiniões.

Por fim deixar uma palavra aos meus amigos por toda a paciência que tiveram ao longo destes meses.

Obrigado!

.

Resumo

Num mundo cada vez mais civilizado e urbanizado as necessidades de construção aumentam exponencialmente. O processo de construção envolve maioritariamente o manuseamento de cargas de grandes dimensões, razão pela qual são frequentemente usadas as gruas e mais recentemente também robôs.

O ambiente de simulação de arquitetura robótica de um "cable robot" proposto nesta dissertação baseia-se num sistema em que as únicas partes móveis são uma plataforma suspensa por um conjunto de quatro cabos. As tensões aplicadas nos cabos provocam a deslocação da plataforma. Este sistema totalmente automatizado tem como principal objetivo a montagem de estruturas de forma rápida, precisa e segura utilizando uma variada gama de ferramentas. Este tipo de robô tem como atributos de destaque em comparação com os robôs convencionais: a sua maior portabilidade, menor custo e possibilidade de construir estruturas maiores.

Nesta dissertação é apresentado todo o processo de desenvolvimento de um ambiente de simulação de um robô de quatro cabos, desde a sua estrutura até ao seu controlo. Este simulador serve de complemento ao protótipo de um robô de cabos desenvolvido na FEUP (Faculdade de Engenharia da Universidade do Porto).

Abstract

In a world increasingly urbanized and civilized construction needs increase exponentially. The construction process mainly involves handling large loads, which is why they are often used cranes and more recently also robots.

The robotic architecture of a simulation environment "cable robot" proposed in this dissertation is based on a system where the only moving parts are a platform suspended by a set of four cables. The tension applied to the cable causing the displacement of the platform. This fully automated system aims to mounting structures quickly, precisely and safely using a range of tools. This type of robot has outstanding attributes as compared with conventional robots: its portability, lower cost and the ability to build larger structures.

In this dissertation is presented the entire process of development of a simulation of a four cables robot environment, from its structure to its control. This simulator complements the prototype of a cable robot developed at FEUP (Faculty of Engineering, University of Porto).

Índice

Lista de acrónimos

FEUP Faculdade de Engenharia da Universidade do Porto

3D Tridimensional

GUI Graphical User Interface ID Identification

PC Personal Computer

Índice de Figuras

1- Introdução

1.1- Contexto e motivação

O aumento da população mundial leva a que as necessidades de infraestruturas, para responder às exigências do mundo atual, cresçam de dia para dia. Os robôs de cabos surgiram por invenção de Khoshnevis [1,2] com a motivação de possibilitar o reposicionamento de elevada amplitude de objetos com grandes dimensões. No seguimento deste tipo de processos surgiu o *Contour Crafting.*

O Contour Crafting permite a construção automatizada de estruturas civis através da sua tecnologia de fabrico por camadas. Esta tecnologia tem como objetivos reduzir o custo da construção e obter melhores níveis de qualidade, segurança e velocidade. À imagem de outras tecnologias de fabrico por camadas, este sistema possibilita a construção de infraestruturas, camada por camada, de baixo para cima, ou seja, dentro do seu volume de trabalho permite a sobreposição de objetos ou qualquer outro material. Ao invés da maior parte deste tipo de tecnologias, este sistema é projetado para estruturas de qualquer escala, tendo melhores desempenhos para a grande escala. Permitem portanto a execução de projetos pequenos como por exemplo casas unifamiliares, mas também de grande envergadura como complexos habitacionais e grandes edifícios destinados a escritórios.

Esta tecnologia utiliza o conceito de robô de cabos. Os robôs de cabos pertencem a uma classe específica de robôs paralelos em que fios substituem os habituais braços rígidos. O funcionamento destes sistemas traduz-se numa menor probabilidade de colisão com objetos que se encontrem na sua área de trabalho, uma vez que os seus braços são menos intrusivos. A sua arquitetura escalar e mecânica simples representam um maior potencial ao nível de volume de trabalho comparativamente com as demais tecnologias. A reduzida necessidade energética para suportar grandes cargas é um fator importante quando comparado com os manipuladores clássicos. Nestes sistemas robóticos a sua plataforma suspensa é deslocada apenas com movimento de cabos, sendo a forma mais simples de o fazer enrolar os cabos com guinchos.

Na arquitetura robótica escolhida para simulação no âmbito desta dissertação são usados quatro cabos. A plataforma que se encontra dentro de uma estrutura é a única ligação entre todos os cabos. Isto resulta de os cabos serem puxados a partir de um

ponto mais elevado (extremos da estrutura) em relação ao centro de massa da plataforma. Ou seja, a força gravitacional é a única força descendente que aplicada em todos os momentos na plataforma. A flexibilidade dos cabos é uma propriedade mecânica a ter em conta, uma vez que estes estão continuamente em tração com diferentes valores de forças. Os três graus de liberdade (posição Cartesiana X, Y, Z) e a variada gama de ferramentas possíveis de aplicar na plataforma concedem ao robô de cabos simulado nesta dissertação a amplitude, força, precisão e versatilidade necessárias para que cumpra os requisitos do sistema.

1.2- Objetivos

O objetivo principal deste trabalho é desenvolver um ambiente de simulação para um protótipo de um sistema robótico "cable robot" desenvolvido na FEUP [14].

Existem atualmente diversos trabalhos de investigação de desenvolvimento de robôs de cabos. Este robô é constituído por uma plataforma (onde se posiciona uma possível ferramenta) ligada por vários cabos (controlados através de motores) que comandam a posição da mesma tridimensionalmente (com vários graus de liberdade).

Comparativamente com um manipulador clássico este sistema tem um custo muito mais baixo, menor inércia e um volume de trabalho muito maior.

Este trabalho consiste em desenhar, projetar e desenvolver um ambiente de simulação de um robô, em que a plataforma é colocada na posição pretendida através de várias linhas devidamente controladas. Isto possibilita, com a adição de uma ferramenta, a execução de tarefas como, por exemplo, *pick-and-place* de objetos.

1.3- Estrutura do documento

Este documento está dividido em 8 capítulos.

No capítulo 2 intitulado de "Estado da arte" são apresentadas várias soluções estudadas quer ao nível protótipos do universo científico quer ao nível de plataformas de simulação. Todas as soluções apresentadas têm pontos em comum com o tema da dissertação.

No capítulo 3 é descrito a estrutura do sistema e seus subsistemas com o intuito de proporcionar uma melhor perceção do projeto realizado.

No capítulo 4 é abordada a componente mais teórica do sistema, mais concretamente a cinemática. Sendo a geometria do robô a base do sistema esta é uma das componentes de maior relevo, o que levou a um estudo de vários modelos de sistema.

O capítulo 5 inclui a implementação prática do sistema desenvolvido no capítulo anterior, com especial incidência na interface com o utilizador e planeamento de trajetórias.

No capítulo 6 é explicado o desenvolvimento da aplicação no *software* de simulação, na sua componente estrutural, comunicação com a interface e simulação do robô idealizado.

O capítulo 7 apresenta os resultados práticos do trabalho realizado nesta dissertação.

Por fim no capítulo 8 são descritas as conclusões retiradas da elaboração do projeto. Este é um capítulo importante, uma vez que, permite uma retrospeção do trabalho realizado e ainda apresenta sugestões para trabalhos futuros de melhoramento do produto apresentado nesta dissertação.

2- Estado da arte

Neste capítulo são abordadas várias soluções que utilizam a tecnologia dos robôs de cabos. Os exemplos presentes abordam o sistema através de diferentes prismas, apesar de conterem arquiteturas semelhantes.

São também analisados vários *softwares* de simulação fazendo uma breve descrição e destacando as suas principais caraterísticas.

2.1-Protótipos

2.1.1- The NIST RoboCrane

O (NIST Nacional Institute os Standards and Technology) experimentou uma grande variedade de aplicações para o seu projeto RoboCrane. As únicas características comuns de todas as abordagens são a utilização de cabos nas ligações paralelas e a utilização de guinchos como atuadores.

Dependendo dos materiais a movimentar existem versões do RoboCrane para terra, ar, água e espaço e têm aplicação por exemplo em: mobilidade de estruturas flexíveis, manuseamento de materiais pesados; apoio a equipas e equipamentos de salvamento e manobra no ar; remoção/levantamento e resgate a partir de referências estáveis ou instáveis em água. Acesso a navios em doca seca, e, veículos lunares de longa distância no espaço.

Todas as versões do RoboCrane abrangem um grande volume de trabalho, seis graus de liberdade, precisão, manobrabilidade e capacidade de guindaste aprimorados [3].

Na figura 2.1 é ilustrado um protótipo de uma versão do RoboCrane para exploração lunar.

Figura 2.1- Protótipo RoboCrane "Lunar Rover" [4]

2.1.2- A Low-Cost Easy Operation 4-Cable Driven Parallel Manipulator, LARM

O objetivo do LARM (Laboratório de Robótica e Mecatrónica de Cassino) é abordar os problemas com vista em aplicações práticas com equipamentos robustos e de fácil operação.

A figura 2.2 mostra um dos seus modelos de robôs de cabos denominado "CALOWI". O sistema de manipulação do "CALOWI" é composto por quatro motores de corrente continua com enroladores acoplados que fazem variar os comprimentos dos cabos. De forma a poder operar em modo plano mas também em tarefas espaciais foi considerado um sistema de transmissão com roldanas. A fim de obter uma estrutura de arquitetura simétrica foi escolhida uma base fixa quadrada para as tarefas em modo plano [5].

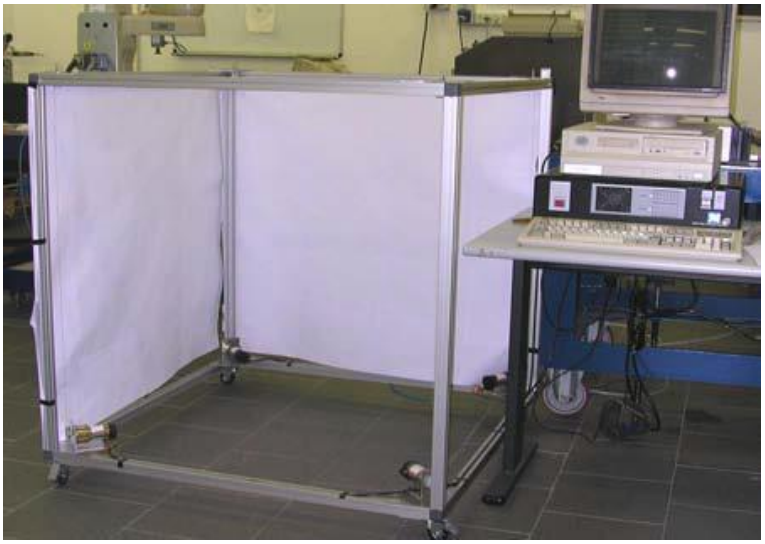

Figura 2.2- Protótipo LARM "CALOWI" [5]

2.1.3- Segesta: The Robot's Net

Na universidade alemã de Duisburg-Essen tem sido desenvolvido por engenheiros e matemáticos do departamento de mecatrónica um robô de cabos chamado Segesta, sendo este inspirado em teias de aranha. Distingue-se dos robôs convencionais pela capacidade de levantar cargas com massa superior com os seus cabos de fibra de alta tecnologia, e tem como vantagem apenas ter de ser considerada a inércia da plataforma e o momento de inércia dos guinchos dos cabos. Estas propriedades fazem com que atinja acelerações até dez vezes superiores à da Terra, e velocidades até dez metros por segundo [6].

Na figura 2.3 é ilustrado um protótipo do Segesta onde se vê a pequena plataforma ao centro junto à base.

Figura 2.3- Protótipo Segesta [7]

2.1.4- IPAnema: A family of Cable-Driven Parallel Robots for Industrial Applications

A família de robôs IPAnema é desenvolvida pelo grupo Fraunhofer IPA desde 2006 para realizar tarefas de média/larga escala de inspeção, manuseamento e montagem. O objetivo principal é produzir um robô baseado em componentes de qualidade industrial e fornecer alta robustez e fiabilidade ao sistema. Por outro lado podem surgir limitações no controlo, uma vez que métodos numéricos de alto nível são normalmente difíceis de integrar em sistemas de tempo real. Posto isto foi assumido um compromisso entre complexidade numérica e robustez, fiabilidade e simplicidade nos componentes de controlo [8].

A Figura 2.4 ilustra um protótipo do IPAnema.

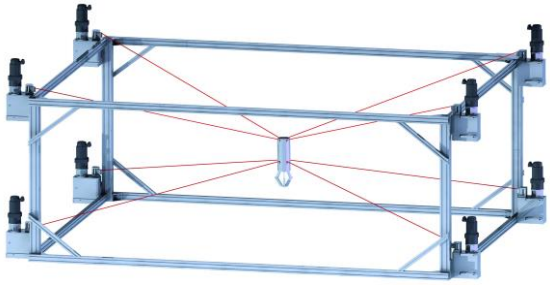

Figura 2.4- IPAnema cable robot [9]

2.2- Plataformas de simulação

2.2.1- V-REP

O v-rep, como desenvolvimento de ambiente integrado, é baseado numa arquitetura de controlo distribuída: cada objeto/modelo pode ser independentemente controlado por um script interno, um plugin, ROS, um cliente API remoto, ou uma solução personalizada. Isto torna o V-REP muito versátil e ideal para aplicações com múltiplos robôs. O controlo pode ser escrito em C/C++, Python, Java, Lua, Matlab, Octave ou Urbi.

Este simulador é utilizado para desenvolvimento rápido de algoritmos, simulação de automações industriais, desenvolvimento de protótipos, robôs educacionais, monitorização remota, etc [10].

Na figura 2.5 é ilustrado um exemplo de um ambiente de simulação desenvolvido no V-REP.

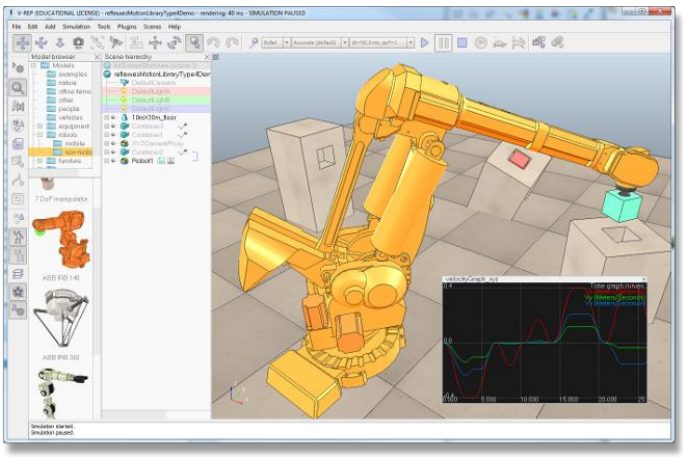

Figura 2.5- Ambiente de simulação em V-REP [10]

As suas principais caraterísticas são:

- Seis focos de programação: O simulador e as simulações são totalmente personalizáveis
- Deteção de colisão e cálculo de distâncias: rápida deteção de interferências e cálculo de distâncias mínimas entre objetos e elementos.
- Planeamento de plataforma

- Interfaces de utilizador personalizáveis: ilimitado número de elementos de interface personalizáveis, com modo de edição integrado.
- Física e dinâmica: cálculos dinâmicos rápidos e personalizáveis para simular a física do mundo real e interação de objetos. Dois modos suportados: Bullet e ODE
- Simulação de sensores de visão: Simulação de sensores de visão com processamento interno de imagem, personalizável via *plugins.*
- Modos de edição integrados: ex. modo editor de malha, incluindo um método semiautomático de extração de formas primitivas (imprescindível para criar cenários com conteúdos dinâmicos otimizados)
- API remoto: controla uma simulação ou o próprio simulador remotamente. Fácil de utilizar, suporta operações sincronizadas ou dessincronizadas, está otimizado para grandes transferências de dados e minimiza as falhas de comunicação. Com 5 linguagens suportadas.
- Simulação de sensores de proximidade: Poderosas simulações com sensores de proximidade totalmente personalizáveis. Desempenha um cálculo de distâncias mínimas exato dentro de uma deteção personalizável de volume.
- Cinemática direta e inversa: cinemática direta e inversa totalmente calculável para qualquer tipo de mecanismo (ramificado, fechado, redundante, etc.)
- Gravação e visualização de dados: uma larga variedade de dados graváveis pode ser usada para mostrar gráficos.

Outras caraterísticas: partículas dinâmicas, simulação de corte de superfícies, gravação de vídeo, simulação de comunicações sem fios, simulação de pintura, espelhos óticos, documentação exaustiva em multinível [10].

2.2.2- SimTwo

O SimTwo surgiu com intuito de conseguir ser um simulador realista principalmente ao nível da dinâmica. A visualização 3D é uma componente importante pois permite uma observação do comportamento do sistema em tempo real. Outro foco é o realismo dos modelos dos motores e respetivos controladores envolvidos. Estes modelos, tentam capturar elementos não lineares que estão invariavelmente presentes nos motores de todos os robôs [11].

O SimTwo é um sistema de simulação realista onde é possível implementar vários tipos de robots:

- Robots móveis com diferentes configurações
    - Diferenciais
    - Com rodas omnidirecionais
- Manipuladores
- Quadrúpedes
- Humanóides
- Qualquer tipo de robot terrestre que possa ser descrito com uma mistura de articulações rotativas e rodas clássicas/omnidirecionais.
- Veículos mais-leves-que-o-ar com ou sem hélices para propulsão

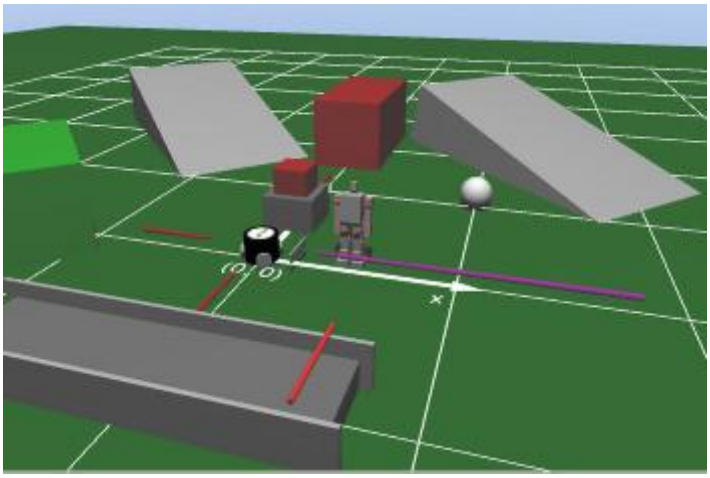

Figura 2.6- Ambiente de simulação em SimTwo [11]

O realismo da dinâmica implementada no SimTwo é conseguida decompondo o robô num sistema de corpos rígidos e motores elétricos. A "mecânica" associada aos corpos é simulada numericamente considerando as suas propriedades físicas: forma, massa e momentos de inércia, atritos das superfícies e elasticidade. Certas articulações típicas: eixo, cardan e calha são definidas explicitamente e podem ter associado um sistema de acionamento e sensores.

O sistema de acionamento pode ser constituído por um motor DC, opcionalmente com caixa redutora e um controlador. O controlador pode ser de vários tipos: um PID aplicado ao sinal de posição ou de velocidade ou uma realimentação de estado.

O modelo do motor DC contém vários elementos não lineares tais como saturação da tensão aplicada, limite de corrente e atrito de Coulomb.

Para além do sistema do controlo de baixo nível, o SimTwo oferece a possibilidade de fornecer os sinais de referência para esses controladores através de um controlador de mais alto nível implementado pelo utilizador. Este controlador pode ser implementado recorrendo:

- A uma linguagem de script editável e compilável no próprio SimTwo.
- A um programa remoto que comunica via rede ou porta série com o SimTwo.

O SimTwo recorre a várias bibliotecas "Open Source":

- GLScene - Para permitir a visualização 3D
- ODE - O motor de simulação de corpos rígidos
- Pascal Script - Para implementar, no SimTwo, um sistema de controlo programável
- SynEdit - Implementa o editor de scripts.
- OmniXml - Facilita o recurso a ficheiros de configuração em formato XML
- RxLib - Conjunto variado de Componentes (Persistência, etc)

2.2.3- Webots

O Webots é um ambiente de desenvolvimento utilizado para modelar, programar e simular robôs móveis.

Com o Webots é possível desenhar cenários robóticos complexos, incluído com robôs diferentes interagindo num ambiente partilhado. As propriedades de cada objeto, tais como forma, cor, textura, massa ou atrito, são individualmente ajustáveis. Uma grande variedade de sensores e atuadores está disponível para equipar cada robô. Os controladores do robô podem ser programado com o IDE integrado ou com qualquer ambiente de desenvolvimento de terceiros. Os comportamentos dos robôs são executados em mundos fisicamente realistas. Os programas de controlado podem, opcionalmente, ser transferidos para robôs reais [12].

Na figura 2.7 é ilustrado um exemplo de um ambiente de simulação desenvolvido em webots.

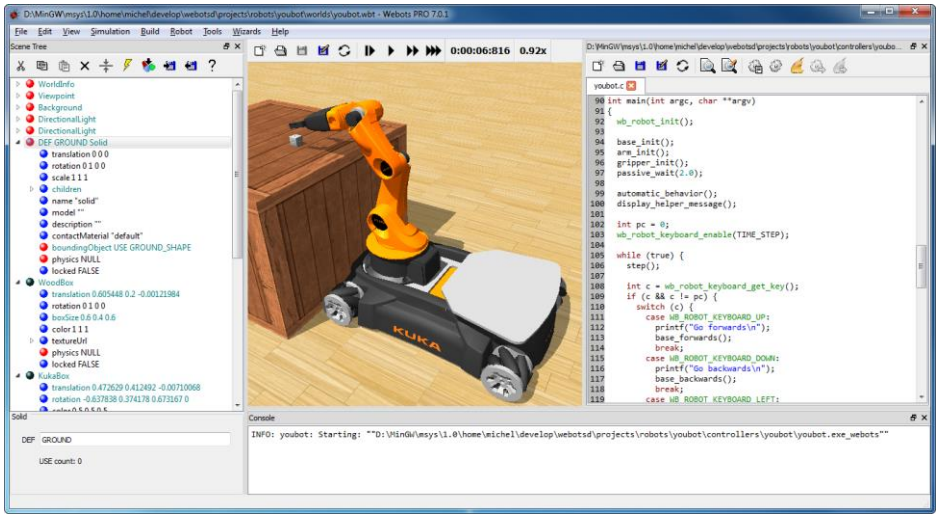

Figura 2.7- Ambiente de simulação em Webots [12]

Principais caraterísticas:

- Modelação e simulação de qualquer robô móvel, incluindo robôs de rodas, pernas e voadores
- Inclui uma biblioteca completa de sensores e atuadores
- Tem um mundo 3D integrado e um editor de robôs em 3D capacidade de importação
- Permite programar os robôs em C / C ++, Python, Java, ROS, Matlab ou através de TCP / IP
- Usa ODE (Open Dynamics Engine) para biblioteca em tempo real simulação de física
- Transferências para robôs reais da: e-puck, NAO, Darwin-OP, pionneer
- Permite simular sistemas multiagentes com facilidades de comunicação.
- Cria filmes HD das simulações.

2.2.4- Gazebo

O Gazebo é um simulador que oferece a capacidade de precisão e eficiência ao simular populações de robôs em ambientes interiores e exteriores complexos. É um motor físico robusto com gráficos de alta qualidade, e convenientes interfaces gráficas e de programação [13].

O Gazebo tem como propósito:

- Simular o mundo real
- Testar e desenvolver *hardware* e *software*
- Testar a regressão
- Estudar
    - Interação humano-robô
    - Simulação física avançada

Na figura 2.8 é ilustrado um exemplo de um ambiente de simulação desenvolvido no Gazebo.

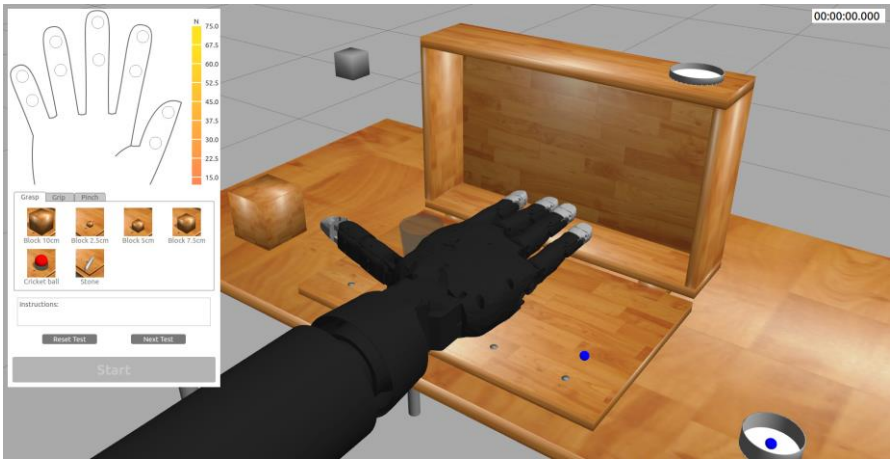

Figura 2.8- Ambiente de simulação em Gazebo [13]

As suas caraterísticas principais são:

- Simulação dinâmica: acesso a múltiplas plataformas de alta performance incluindo ODE, Bullet, Simbody e DART
- Gráficos 3D avançados: utilizando OGRE, proporciona ambientes realistas, incluindo iluminação de alta qualidade, sombras e texturas.
- Sensores e ruído: gera dados dos sensores, opcionalmente com o ruido, a partir de lasers de deteção de alcance, câmara 2D/3D, sensores estilo Kinect, sensores de contato, força de tração, etc
- Plugins: desenvolver plugins personalizados para robôs, sensores e ambientes. Fornece acesso direto à API do Gazebo.

- Modelos de robôs: inclui modelos de robôs PR2, Pioneer2 DX, iRobot Create e TurtleBot.
- Comunicação TCP/IP: executa simulações em servidores remotos, com interface para o Gazebo através do Google Protobufs
- Linha de comandos: ferramentas de linha de comandos extensas para facilitar a introspeção, simulação e controlo.

3- Arquitetura

O projeto iniciou-se pela fase de planeamento e análise de requisitos. Neste capítulo são abordadas as bases do projeto, a escolha do modelo e da estrutura. Seguidamente deu-se início á construção do sistema de simulação.

3.1- Estrutura

O objetivo principal desta dissertação é a realização de um ambiente de simulação de um robô de cabos para um protótipo desenvolvido na FEUP [14]. Tendo em conta a implementação em ambiente real com o respetivo *hardware* foi planeado de forma a usar uma arquitetura e modelo de baixos recursos. Um fator importante é o número de cabos a utilizar, que deve ser o mais reduzido possível mas sem descurar os requisitos do sistema.

Sendo os principais fatores o volume de trabalho, grau de liberdade, precisão, entre outros, foi escolhida uma arquitetura de quatro cabos superiores ao centro de massa da ferramenta central de forma a combinar adequadamente todos estes fatores. A estrutura a utilizar segundo o protótipo desenvolvido é um paralelepípedo de dimensões 120x60x60 centímetros. As quatro roldanas que puxam os cabos estão fixas nos vértices superiores da estrutura. A plataforma é retangular com dimensões 2x1x1 centímetros sendo esta a única ligação dos quatro cabos respetivos a cada vértice da estrutura. Na Figura 3.1 é ilustrado um modelo 3D do robô de cabos descrito.

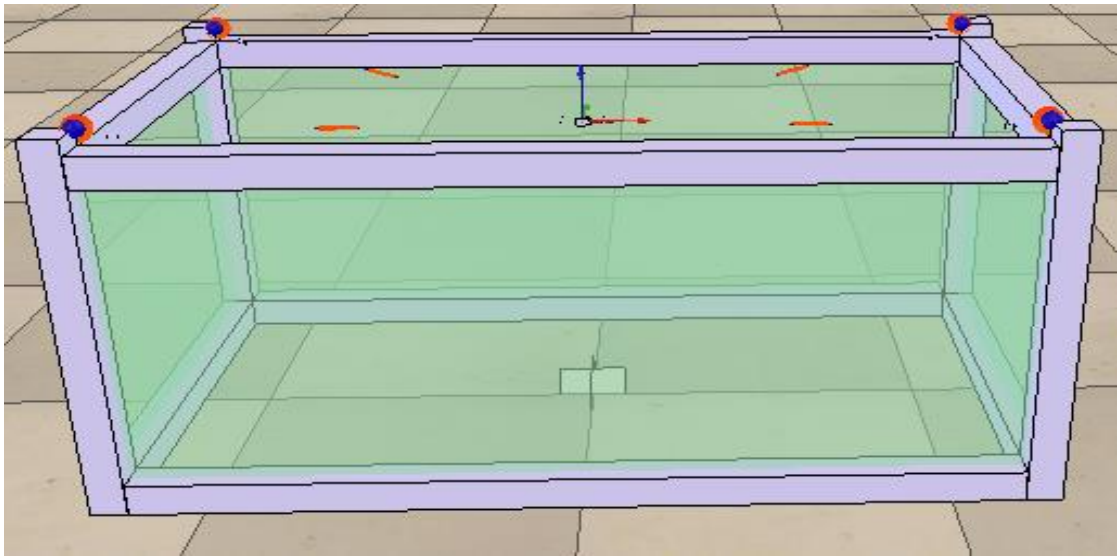

Figura 3.1- Modelo 3D do robô de cabos

Esta arquitetura robótica apresenta três graus de liberdade (coordenadas cartesianas X,Y,Z), já que, dadas as reduzidas dimensões da plataforma, pode desprezar-se os ângulos Roll e Pitch da mesma.

Com este tipo de arquitetura a única força descendente sempre presente é a força gravitacional, uma vez que os cabos estão fixos acima do centro de massa da plataforma.

Este sistema tem como principal requisito o posicionamento paralelo dos objetos em relação ao tampo inferior da estrutura. Seguidamente é ilustrada nas Figuras 3.2, 3.3 e 3.4 a estrutura vista de vários ângulos.

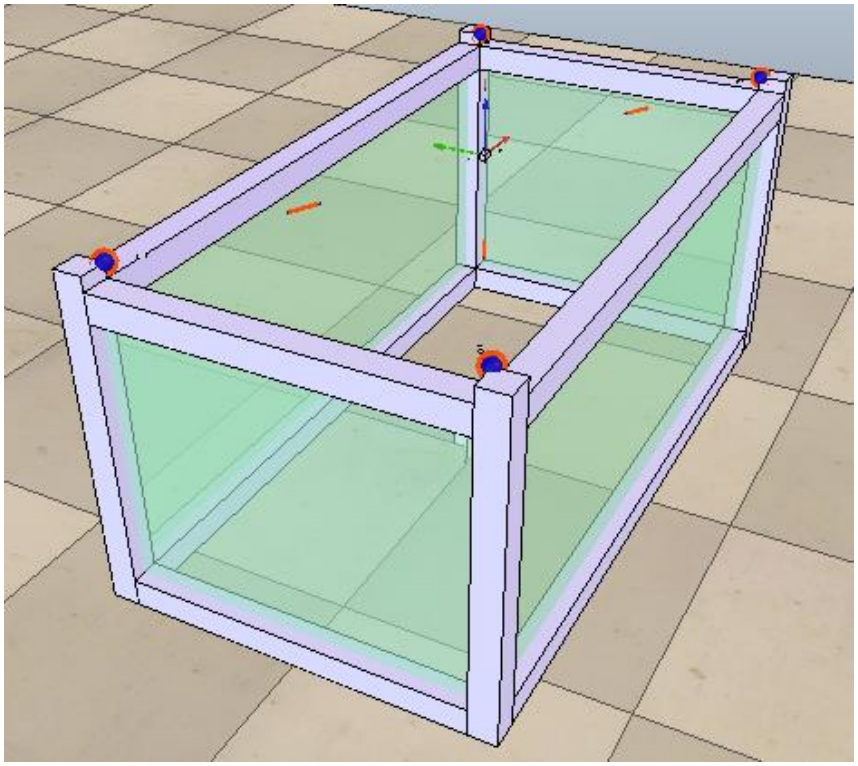

Figura 3.2 - Modelo 3D do Robô- Vista SO

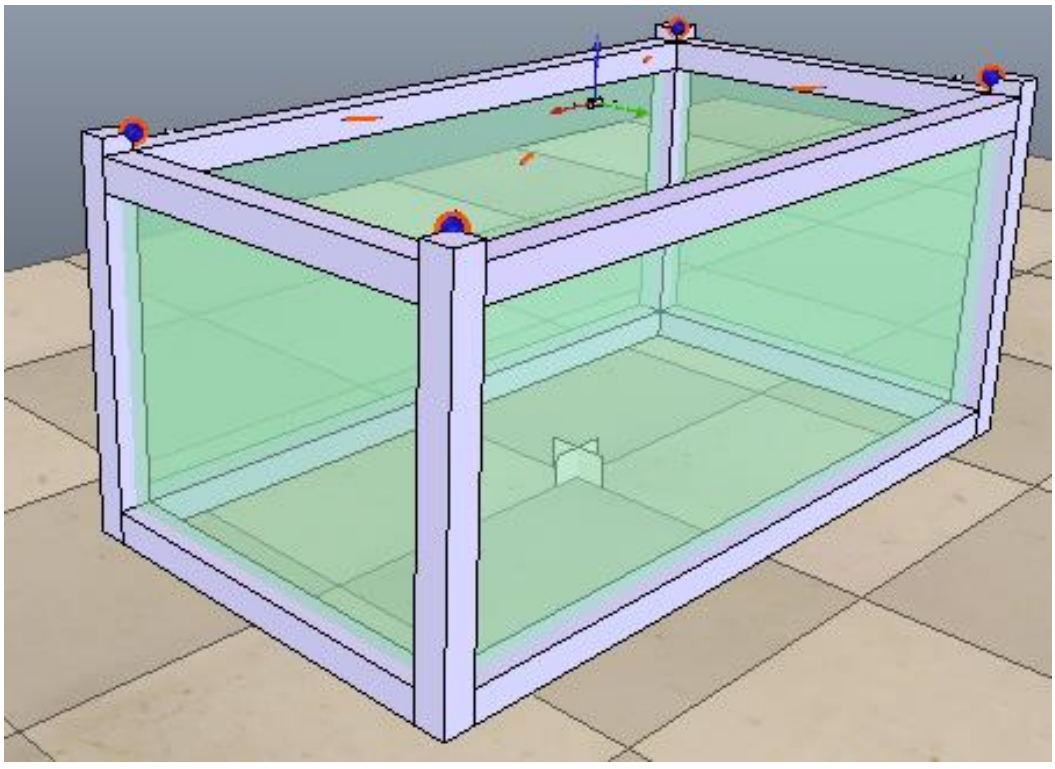

Figura 3.3- Modelo 3D do Robô- Vista SE

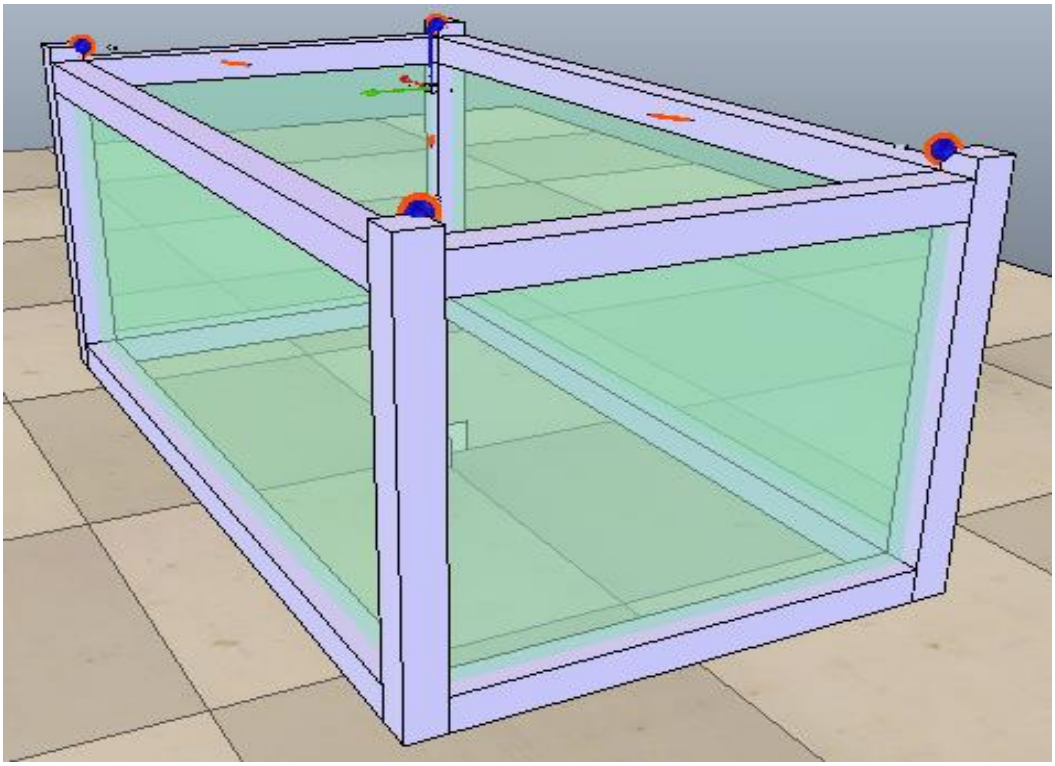

Figura 3.4- Modelo 3D do Robô- Vista NE

3.2- Subsistemas

O sistema de implementação é composto por dois elementos distintos que interagem entre si num computador (PC) ou através de uma rede entre dois ou mais computadores (PC's). O primeiro elemento consiste numa aplicação de controlo desenvolvida no Lazarus, o segundo elemento é o ambiente de simulação desenvolvido no *software* v-rep.

A aplicação de controlo é responsável por:

- Interface com o utilizador
- Processamento das variáveis
- Calculo dos comprimentos dos cabos
- Comunicação dos valores de saída com o V-Rep

O ambiente de simulação no *software* V-Rep é responsável por:

- Receção dos dados inseridos pelo utilizador no Lazarus
- Processamento dos dados

- Simulação do sistema

O sistema é programado pelo utilizador através de uma interface gráfica GUI (Graphical User Interface) na qual pode fornecer informação ao sistema. A posição pretendida para a plataforma central é definida nesta interface através de dados em forma de coordenadas X,Y,Z.

As únicas partes móveis do robô são a plataforma e os cabos que a puxam e são os únicos responsáveis pelo posicionamento e deslocamento. Posto isto, as variáveis do sistema são os comprimentos dos 4 cabos, uma vez que, influenciam diretamente todos os seus graus de liberdade.

Após serem inseridas as coordenadas pelo utilizador, o sistema calcula o comprimento dos cabos conforme a posição de destino. Depois de recebida a informação do comprimento dos cabos no V-rep é necessário fazer com que estes sejam injetados no sistema.

Os comprimentos dos cabos são manipulados através da diferenciação numérica entre os valores de referência e os valores das variáveis do sistema (comprimento real pretendido). O esquema da Figura 3.5 representa a interação entre as partes do sistema geral.

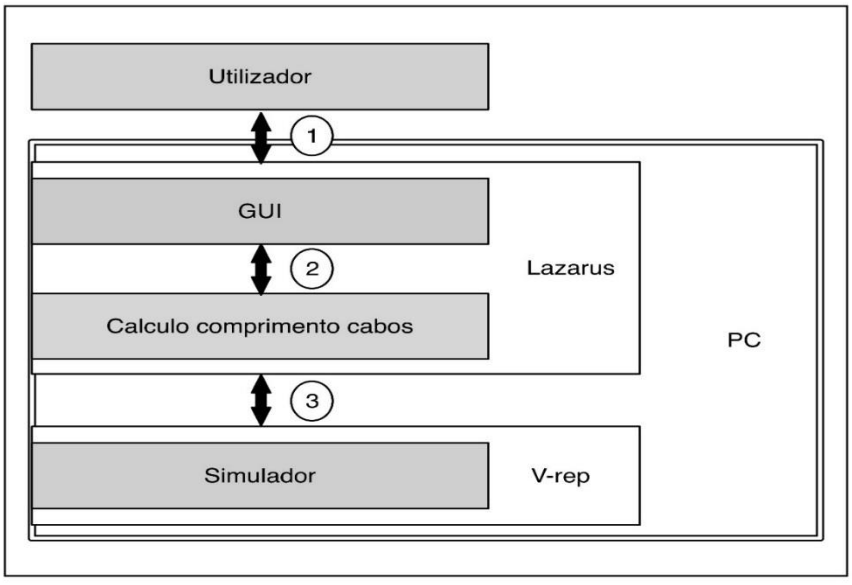

Figura 3.5- Esquemático do sistema

As interações entre as várias partes do sistema estão representadas pelas setas numeradas de 1 a 3 no esquema da Figura 3.5.

Para uma mais fácil perceção do sistema utilizado, seguidamente são analisadas as interações referidas:

1. Ligação entre o Utilizador e o Sistema. Permite introduzir as posições desejadas para a plataforma, monitorizar o comprimento dos cabos. É realizada através de uma interface gráfica, GUI (Graphical User Interface), executável em PC;
2. Para que seja executado um movimento da plataforma é necessário calcular o comprimento de cada um dos quatro cabos. Para isso são importados do GUI para a aplicação de controlo os valores da posição cartesiana inserida pelo utilizador.

3. Rede de comunicação entre a aplicação de controlo e o v-rep (simulador). Depois de calculados os comprimentos dos cabos, são enviados para o simulador que os compara com valores de referência e executa o movimento da plataforma. A rede utilizada tanto pode ser interna (no mesmo PC) ou externa (por wi-fi ou cabo entre vários PC's)

4- Modelo do sistema

4.1- Cinemática

Para controlar uma estrutura robótica foi necessário realizar o estudo geométrico dos vários graus de liberdade que a compõe. Uma vez que o objectivo é simular o funcionamento do protótipo real já existente, foi utilizado para esta dissertação um estudo já realizado para o mesmo. Neste subcapítulo é demonstrado o estudo das duas cinemáticas para várias versões de robôs de quatro cabos.

4.1.1- Modelo simplificado [14].

O ângulo e comprimento de cada cabo é determinante para o cálculo da posição do ponto móvel, tendo em conta também as variáveis das junções dos extremos e do ponto central. Uma vez que um dos requisitos é que os cabos estejam sempre em tensão, a cinemática deste sistema é equivalente á dos robôs de atuadores paralelos. Seguidamente é apresentado o exemplo mais simples da análise do problema, na versão no plano de um robô de quatro cabos quanto á cinemática inversa e direta. A Figura 4.1 ilustra o modelo simplificado utilizado.

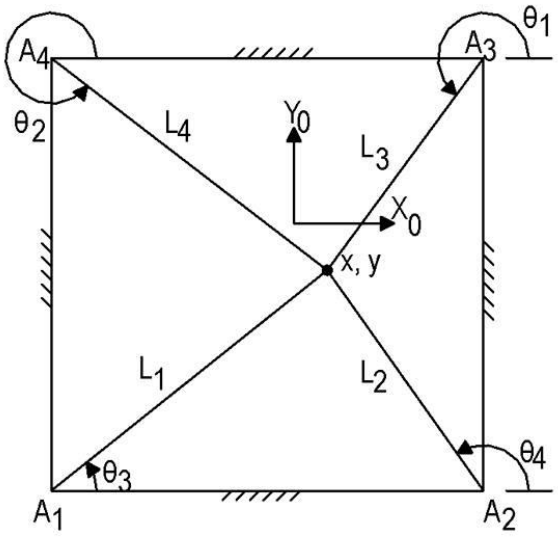

Figura 4.1- Modelo simplificado [14]

A norma Euclidiana entre a posição do ponto móvel $P = \{x\ y\}^T$ e os pontos fixos de cada vértice (A1, A2, A3 e A4) permite resolver o problema da cinemática inversa. Isto permite calcular os comprimentos dos cabos (L1, L2, L3 e L4):

$$Li = \sqrt{(x - Aix)^2 + (y - Aiy)^2}, \quad i = 1, \dots, 4$$

(4.1)

A posição do ponto móvel é calculada através da cinemática direta, dados os comprimentos dos cabos Li restritos a valores consistentes.

Relativamente aos cabos 1 e 2, deslocar a origem do referencial para A1, $A1 = \{0\ 0\}^T$ e $A2 = \{L_b\ y\}^T$, simplifica o problema. O resultado da cinemática direta é dado pela interseção de dois círculos, um com centro em A1 de raio L1, e outro centrado em A2 de raio L2:

$$x = \frac{LB^2 + L1^2 - L2^2}{2 * LB}$$

(4.2)

$$y = \pm\sqrt{L1^2 - x^2}$$

(4.3)

Para garantir que se mantem dentro dos limites do polígono é escolhida a solução positiva da equação de y (4.3). Para as múltiplas combinações de cabos existe apenas uma solução correta. O valor de x na equação (4.2) é único, uma vez que para os cabos 1 e 2 a sua geometria específica resulta em valores de x iguais com valores de y diferentes.

O resultado final depende da deslocação do referencial para a sua posição original. Este método pode ser aplicado no plano para qualquer robô de n cabos.

4.1.2- Modelo simplificado com plataforma retangular [14].

Adaptando alguns conceitos e dimensões da análise da cinemática do caso anterior é possível o estudo de um modelo mais complexo com forma retangular no plano. Na Figura 4.2 é apresentado um modelo simplificado em 2D com plataforma.

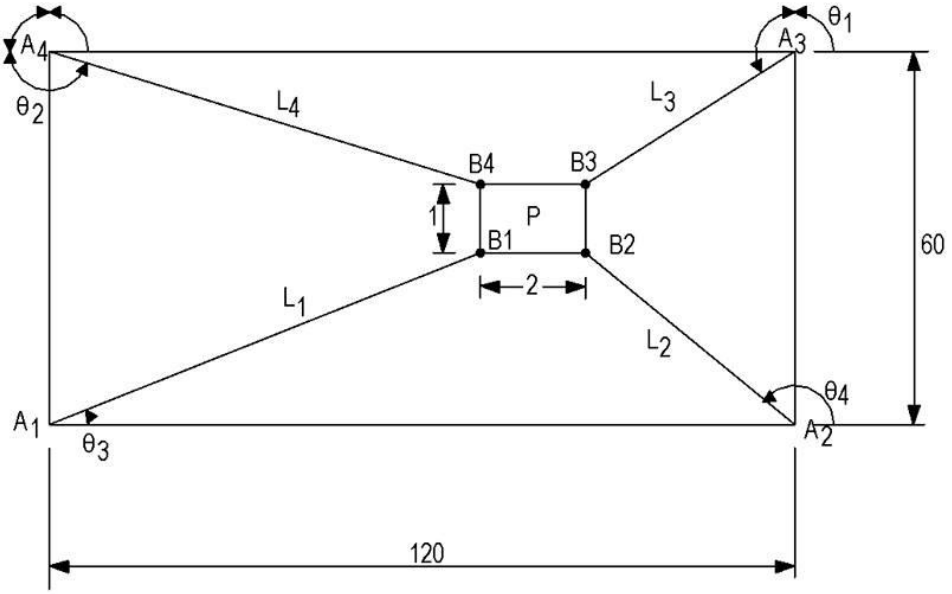

Figura 4.2- Modelo simplificado com plataforma [14].

Na Figura 4.2 é representado o modelo utilizado mas na versão plana. A origem do referencial Cartesiano coincide com o ponto A1. Como referido anteriormente a estrutura tem como dimensões 120x60 centímetros e a plataforma central 2x1 centímetros.

A cinemática inversa neste problema é idêntica à do caso anterior com a diferença que neste existem quatro pontos móveis (B1, B2, B3, B4).

Uma vez que está em estudo a versão no plano XY (Z=0) então B1x = B4x; B2x = B3x; B1y = B2y; B3y = B4y. Logo o ponto $P = \{x\ y\}^T$ está sempre na mesma posição relativamente aos pontos coincidentes com os vértices da plataforma.

Usando mais uma vez a norma Euclidiana entre a posição de cada ponto da plataforma ($Bi = \{x\ y\}^T, i = 1, \dots, 4$) e cada ponto fixo coincidente com os vértices da estrutura ($Bi = \{x\ y\}^T, i = 1, \dots, 4$) é possível calcular os comprimentos dos cabos. Sendo o ponto central da plataforma (P) o que interessa calcular, calculou-se as coordenadas dos pontos Bi através da posição de P relativamente á origem do referencial respeitando as dimensões da plataforma:

$$Bix = Px - \frac{comprimento\ da\ plataforma}{2},\ \ i = 1, \dots, 4$$

(4.4)

$$Biy = Py - \frac{largura\ da\ plataforma}{2},\ \ i = 1, \dots, 4$$

(4.5)

O comprimento de cada cabo é dado por:

$$Li = \sqrt{(Bix - Aix)^2 + (Biy - Aiy)^2},\ \ i = 1, \dots, 4$$

(4.6)

4.1.3- Modelo simplificado 3D com plataforma retangular [14].

Para fazer a análise no espaço adiciona-se um eixo ao referencial Cartesiano ficando com X, Y e Z mantendo a origem coincidente com o ponto A1.

$$A1_{(x,y,z)} = (0,0,0)$$

$$A2_{(x,y,z)} = (120,0,0)$$

$$A3_{(x,y,z)} = (120,60,0)$$

$$A4_{(x,y,z)} = (0,60,0)$$

As dimensões da estrutura passam a ser 120x60x60 centímetros e da plataforma 2x1x1 centímetros. Na Figura 4.3 é ilustrado o modelo simplificado em 3D.

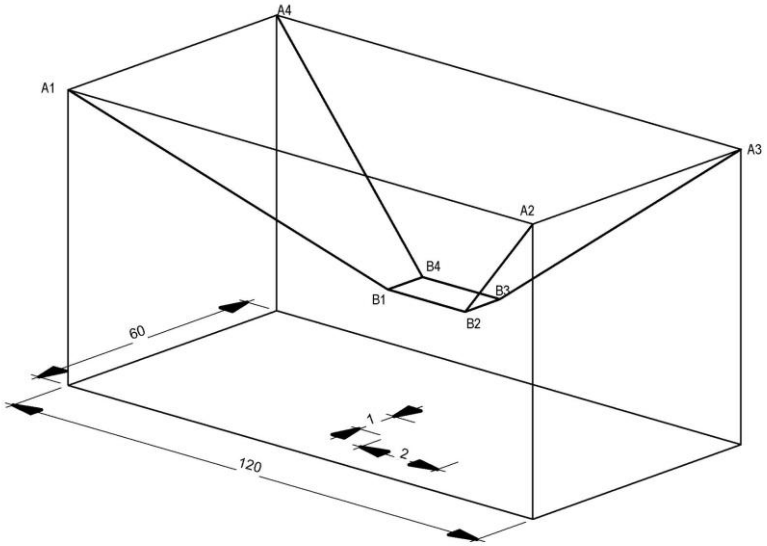

Figura 4.3- Modelo simplificado 3D com plataforma [14].

Para este modelo é necessário uma análise por fases, começando por assumir que B1x = B4x; B2x = B3x; B1y = B2y; B3y = B4y, Biz = B2z = B3z = B4z, mantendo a plataforma paralela ao plano XY.

Acrescentando a coordenada dos ZZ ao sistema anterior pode calcular-se as coordenadas dos pontos móveis onde são fixados os quatro cabos na plataforma, sendo o ponto central da mesma o que interessa calcular:

$$Bix = Px - \frac{comprimento\ da\ plataforma}{2}, \quad i = 1, \dots, 4 \tag{4.7}$$

$$Biy = Py - \frac{largura\ da\ plataforma}{2}, \quad i = 1, \dots, 4 \tag{4.8}$$

$$Biz = Pz, \qquad i = 1, \dots, 4 \tag{4.9}$$

O comprimento de cada cabo é dado por:

$$Li = \sqrt{(Bix - Aix)^2 + (Biy - Aiy)^2 + Biz\text{^}2}, \quad i = 1, \dots, 4 \tag{4.10}$$

5- Implementação

5.1- Trajetória e planeamento

A movimentação da plataforma, dentro do espaço de trabalho, de um ponto para outro não deve ser feita de forma direta e abrupta. O seu deslocamento deve comportar pontos intermédios.

Os destinos intermédios são essenciais para que a movimentação não sobrecarregue a tensão de nenhum dos cabos. Será o utilizador a definir o número de pontos intermédios que pretende para cada movimento.

Ter uma trajetória dividida em vários pequenos percursos não resulta numa maior distancia percorrida entre o ponto de partida e o destino uma vez que o trajeto é o mesmo.

5.2- Interface

Durante o desenvolvimento deste projeto foram necessários conhecimentos em várias áreas: matemática, geometria, cinemática estática de robôs, entre outro. No entanto um futuro utilizador deste simulador não precisa de dominar nenhuma destas áreas podendo apenas utilizar o interface básico e intuitivo desenvolvido para esta aplicação. A Figura 5.1 ilustra o interface gráfico desenhado para este sistema.

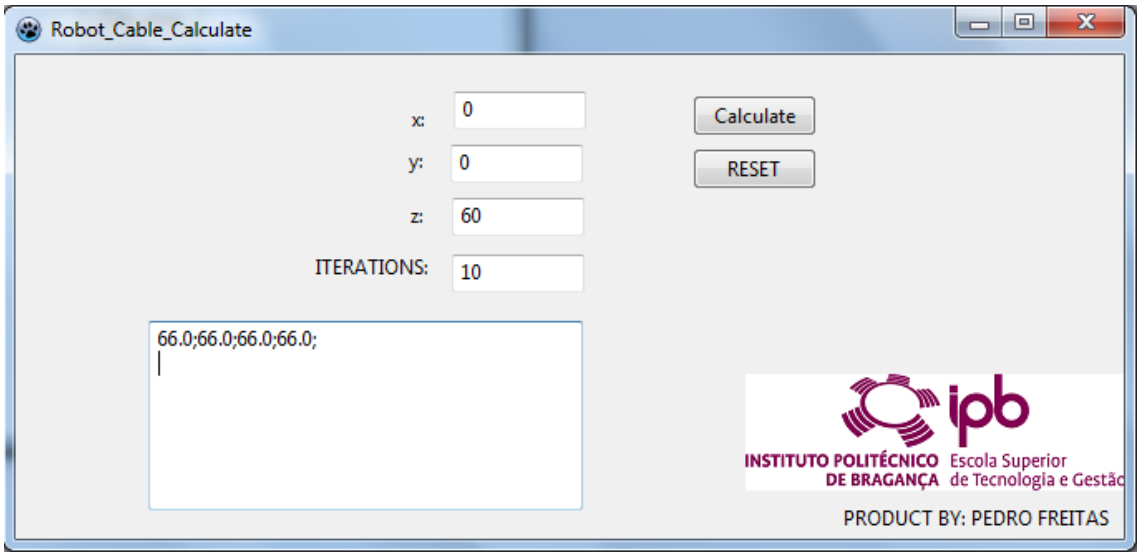

Figura 5.1- Interface gráfico

5.2.1- Iniciação e reiniciação

Aquando da primeira utilização do robô o sistema está programado por segurança para que o mesmo se encontre na posição de origem (x=0, y=0, z=60) sendo o ponto de referência o centro inferior da estrutura (x=0, y=0, z=0).

Sempre que o utilizador pretender pode reiniciar o sistema, utilizando para isso o botão "RESET" existente na interface (ver na Figura 5.2), fazendo o robô retomar a posição de origem com o trajeto dividido por defeito em cinquenta iterações.

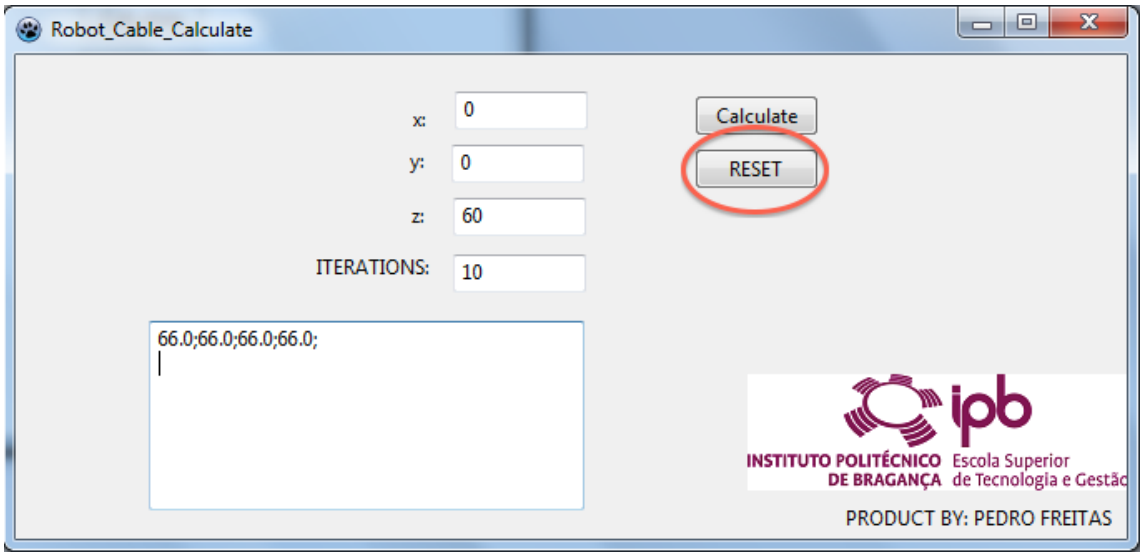

Figura 5.2- Pormenor do botão "reset" do interface

5.2.2- Uso do sistema

A interface com o utilizador é realizada através um *software* com ambiente gráfico (GUI- Graphical User Interface), tratando-se de um programa executável (extensão ".exe"). O grafismo facilita e torna intuitiva a utilização do simulador do robô. O programa permite ainda visualizar o comprimento de cada cabo (ver Figura 5.3), informação que será transmitida ao simulador automaticamente.

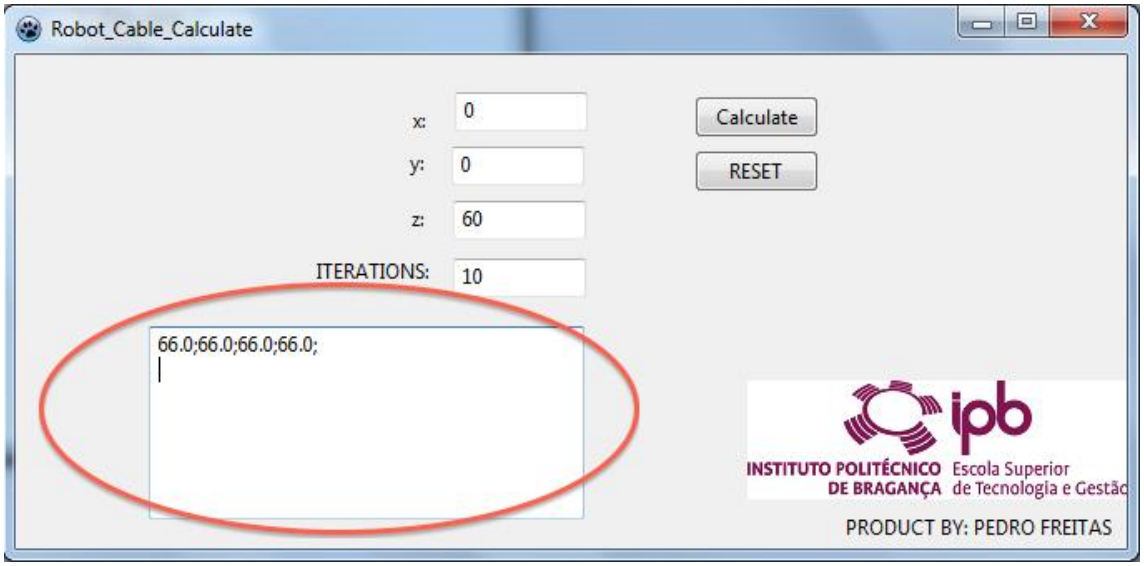

Figura 5.3- Pormenor da indicação do comprimento dos cabos no interface

Como referido anteriormente o robô é capaz de fazer deslocar a sua plataforma móvel dentro da área de trabalho. De acordo com as limitações do robô, ao utilizador basta definir o ponto cartesiano que pretende como destino. Para isso no interface existem três campos editáveis na qual devem ser inseridas as coordenadas (x, y, z) pretendidas, bem como o número de iterações desejado. Depois de inserir os valores pretendidos basta clicar no botão "calculate". (ver Figura 5.4)

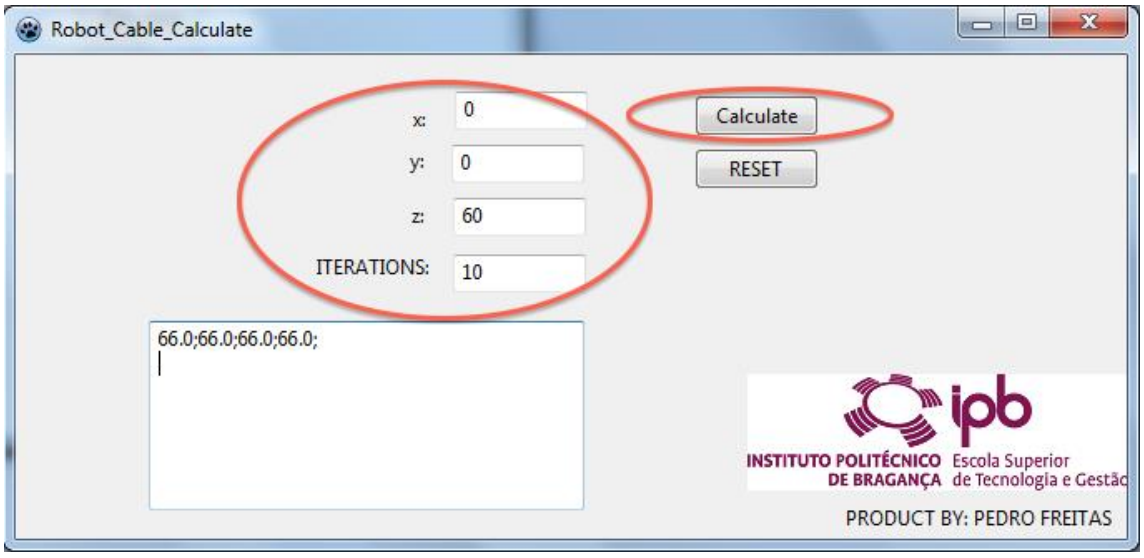

Figura 5.4- Pormenor dos campos dos valores cartesianos e do botão "calculate" no interface

6- Simulador

O ambiente de simulação do robô foi implementado no *software* v-rep (virtual robot experimentation platform) anteriormente analisado no estado da arte..

6.1- Estrutura

A estrutura desenvolvida neste simulador é composta por uma armação em aço, imóvel com cinco centímetros de espessura pousada no solo, com dimensões interiores de 120x60x60 centímetros e proteção em acrílico nas laterais. A plataforma móvel tem dimensões 2x1x1 centímetros. A figura 6.1 ilustra a estrutura no simulador.

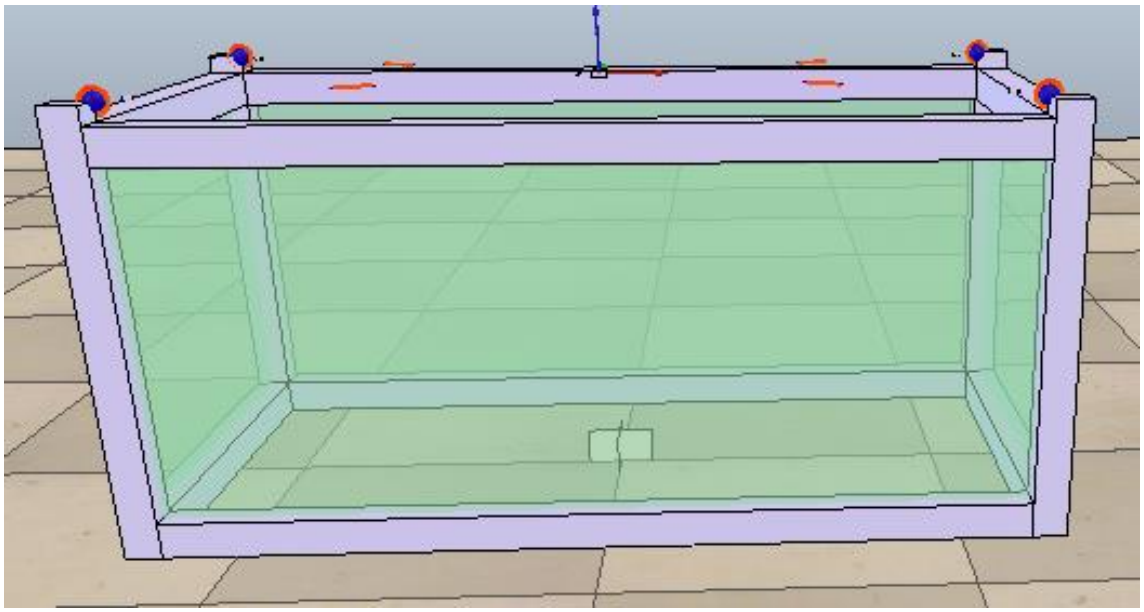

Figura 6.1- Estrutura no simulador

Nos quatro vértices superiores da estrutura e da plataforma encontram-se juntas esféricas nas quais estão acoplados os cabos (ver Figura 6.2). Neste simulador não é possível implementar visualmente cabos, sendo substituídos por juntas extensíveis. Por impedimento do funcionamento do simulador não é possível ligar uma junta a outra, havendo necessidade de usar elementos de ligação.

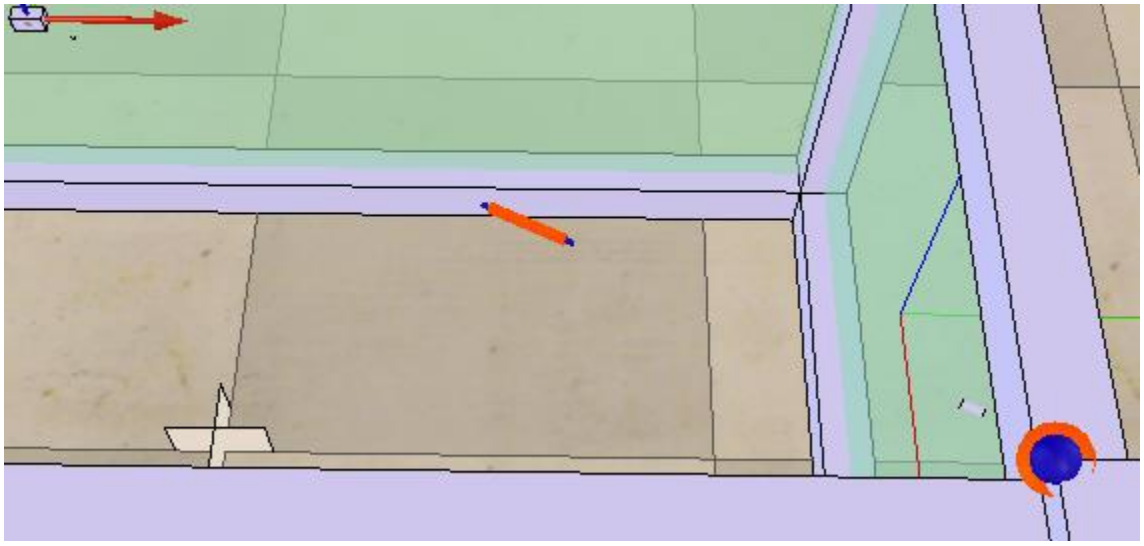

Figura 6.2- Pormenor das juntas esféricas nos vértices

Através de um sistema de hierarquias (ver Figura 6.3), a plataforma encontra-se no topo de uma cadeia de elementos. Seguem-se depois as juntas esféricas da plataforma, elementos de ligação, juntas extensíveis, elementos de ligação, juntas esféricas da estrutura e por fim o pilar da armação.

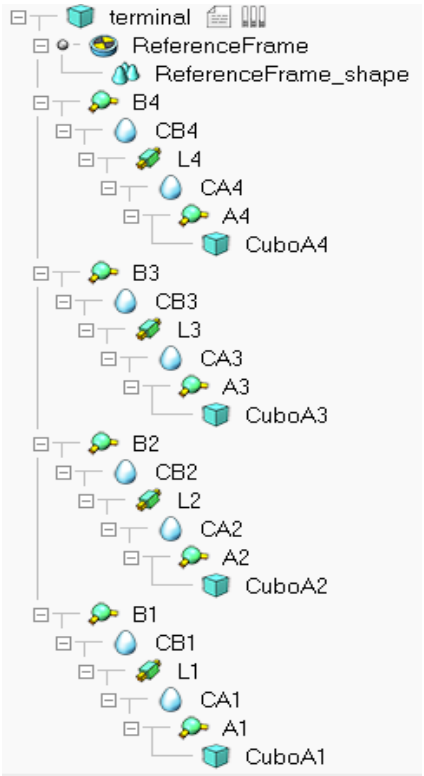

Figura 6.3- Hierarquias dos elementos do simulador

Sempre que existir movimentação da plataforma todos os restantes elementos reagem correspondentemente, á exceção do pilar da armação que apenas está incluído porque uma junta não pode ser o fim da cadeia de elementos.

A movimentação é controlada através de um script desenvolvido para esta aplicação.

Uma vez que a massa dos cabos e da plataforma é muito reduzida os seus valores reais são desprezados ao nível do controlo da dinâmica do sistema, sendo atribuído automaticamente um valor residual relativo aos materiais utilizados nos vários elementos no simulador.

6.2- Comunicação e aquisição de dados

Depois de dar início á simulação é aguardado um envio de dados pela aplicação de controlo através da rede por UDP na porta configurada (ver Figura 6.4).

```
1  if (simGetScriptExecutionCount()==0) then
2
3
4
5      -----------  LAZARUS ----------------
6
7  local socket = require("socket")
8  udp = assert(socket.udp())
9  udpRec = assert(socket.udp())
10 assert(udpRec:setsockname ('*', 8278))
11 udpRec:settimeout(0)
12
13 end
```

Figura 6.4 - Fração do script relativa a comunicação entre aplicação de controlo e simulador

Os comprimentos dos cabos calculados são enviados através de uma string. Quando o v-rep recebe a string divide-a por cabo através do delimitador “;”, associando o valor de cada cabo á respetiva variável ao mesmo tempo que mostra os valores na barra de estados (ver figura 6.5).

```
19 local data, ip, port = udpRec:receivefrom()
20
21 if(data ~= nil) then
22
23     function split(s, delimiter)
24         result = {};
25     for match in (s..delimiter):gmatch("(.-)"..delimiter)
26         table.insert(result, match);
27     end
28     return result;
29 end
30
31 vetor = split(data, ";");
32
33 cabo1=vetor[1];
34 cabo2=vetor[2];
35 cabo3=vetor[3];
36 cabo4=vetor[4];
37
38     simAddStatusbarMessage(cabo1)
39     simAddStatusbarMessage(cabo2)
40     simAddStatusbarMessage(cabo3)
41     simAddStatusbarMessage(cabo4)
```

Figura 6.5- Fração do script relativa a receção e processamento dos dados

De seguida o valor recebido de cada cabo é comparado com o comprimento dos cabos quando a plataforma se encontra na posição de origem (x=0, y=0, z=0.6 metros) e é feito o acerto de unidades uma vez que na aplicação de controlo desenvolvida a unidade utilizada é centímetros (ver Figura 6.6).

```
43         c1=(cabo1-(65.96))/100;
44         c2=(cabo2-(65.96))/100;
45         c3=(cabo3-(65.96))/100;
46         c4=(cabo4-(65.96))/100;
```

Figura 6.6- Fração do script relativa a conversão de unidades

6.3- Simulação

Após a aquisição e processamento dos dados, é associado o valor final de cada cabo á respetiva junta extensível. Como o ponto de referência é o centro inferior da estrutura os cabos 2 e 3 assumirão valores negativos por defeito (ver Figura 6.7).

```
50      joint1=simGetObjectHandle('L1')
51          position=simSetJointTargetPosition(joint1,c1);
52
53
54      joint2=simGetObjectHandle('L2')
55          position=simSetJointTargetPosition(joint2,-c2);
56
57
58      joint3=simGetObjectHandle('L3')
59          position=simSetJointTargetPosition(joint3, -c3);
60
61      joint4=simGetObjectHandle('L4')
62          position=simSetJointTargetPosition(joint4, c4);
```

Figura 6.7- Fração do script relativa a ordem de trabalho

7- Resultados práticos

Neste capítulo são apresentados os resultados de alguns testes básicos ao simulador.

O primeiro teste apresentado (ver Figura 7.1) consiste em movimentar a plataforma desde a sua origem xyz (0; 0; 60) até ao fundo da estrutura e limite do eixo Z com as coordenadas XYZ (0; 0; 0) de forma a que seja visível o movimento da plataforma ao longo de apenas um eixo.

No segundo teste (ver Figura 7.2) é demonstrado a sequência de movimentos da plataforma desde a sua origem xyz (0; 0; 60) até ao limite positivo do eixo X com as coordenadas XYZ (60; 0; 60) de forma a que seja visível o movimento da plataforma ao longo de apenas um eixo..

No terceiro teste (ver Figura 7.3) é demonstrado a sequência de movimentos da plataforma desde a sua origem xyz (0; 0; 60) até ao limite positivo do eixo Y com as coordenadas XYZ (0; 30; 60) de forma a que seja visível o movimento da plataforma ao longo de apenas um eixo..

No quarto teste (ver Figura 7.4) é demonstrado a sequência de movimentos da plataforma desde a sua origem XYZ (0; 0; 60) até ao ponto intermédio dos vários eixos com as coordenadas XYZ (-30; -15; 30) de forma a que seja visível o movimento da plataforma, bem como a resposta dinâmica de todo o sistema a um movimento ao longo dos três eixos dos seus graus de liberdade em simultâneo.

Estes testes básicos permitirão entender mais rápida e simplesmente o ambiente de simulação desenvolvido ao longo deste projeto.

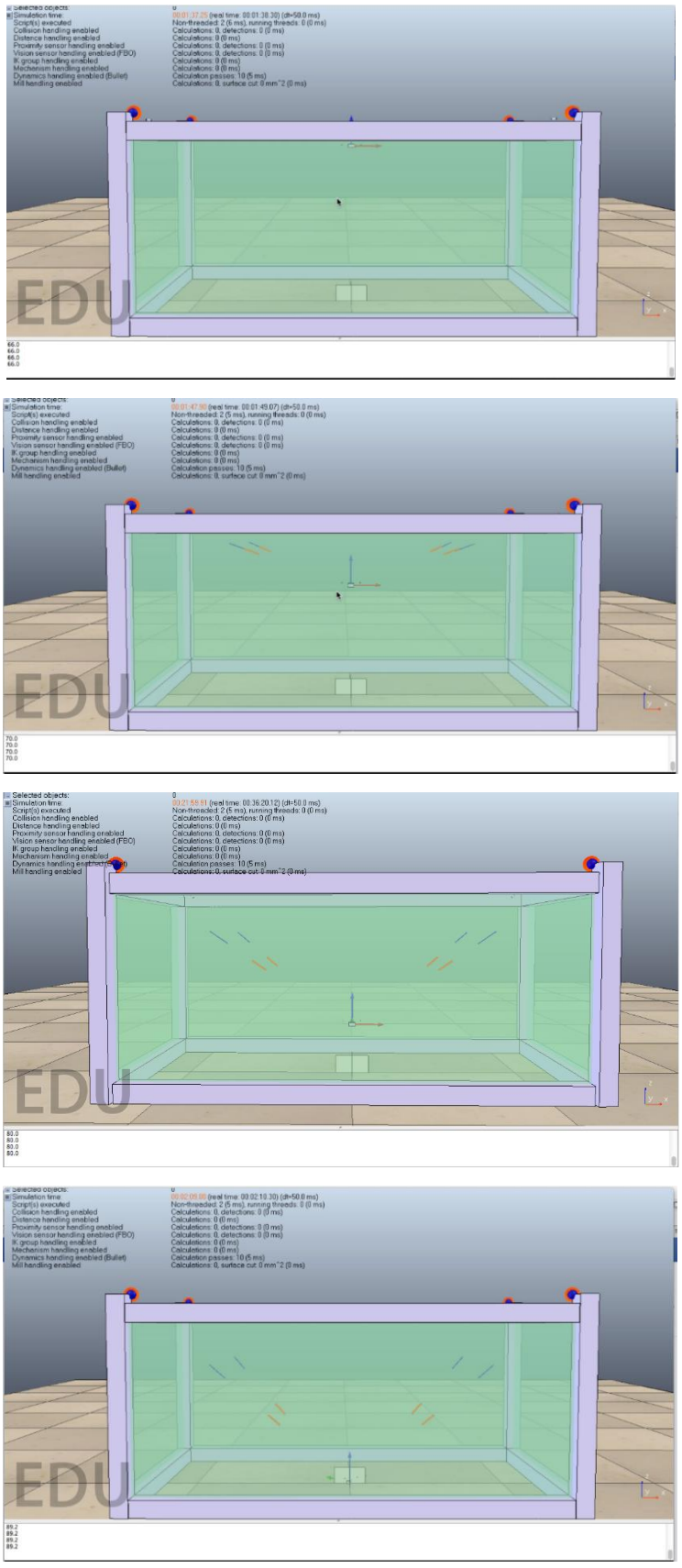

Figura 7.1- Sequência do movimento no eixo de Z

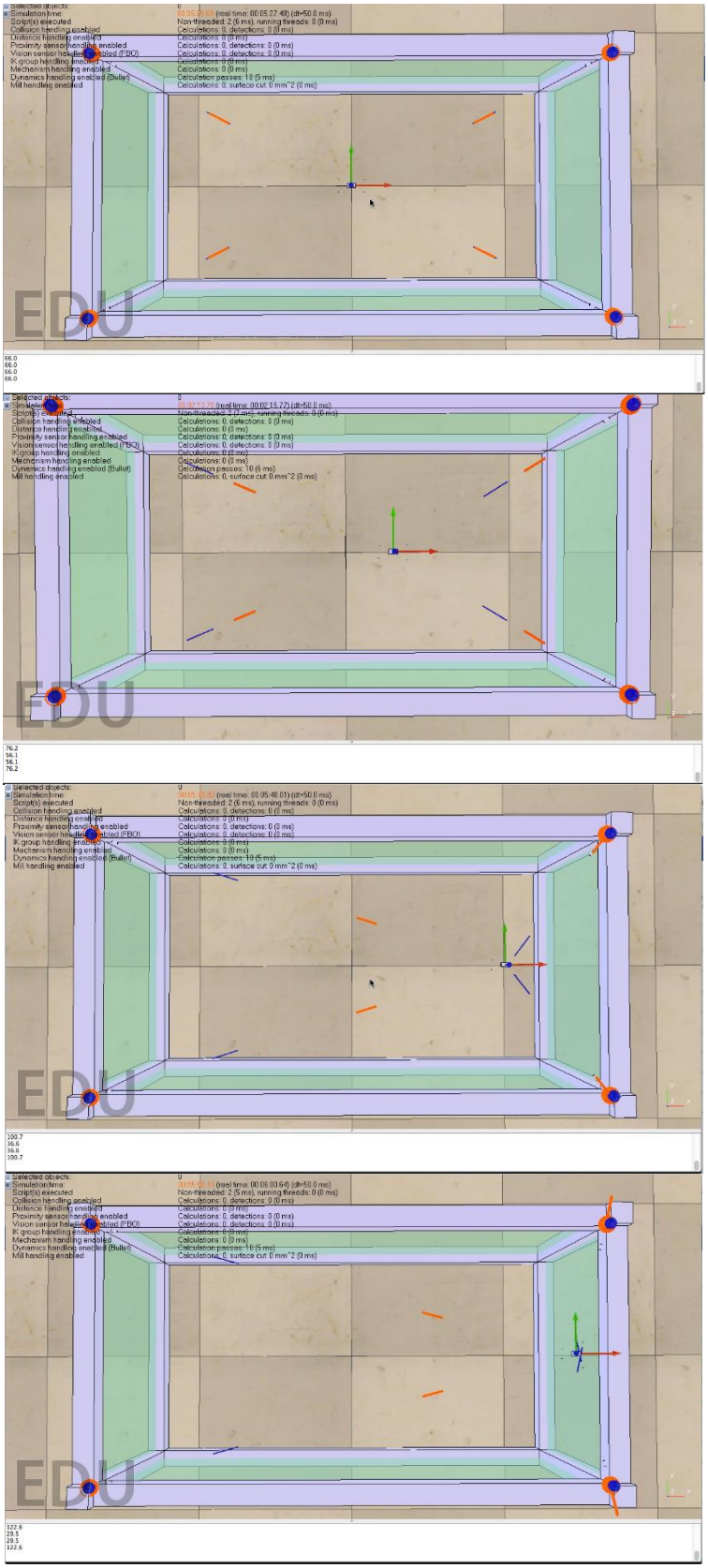

Figura 7.2- Sequência do movimento no eixo de X

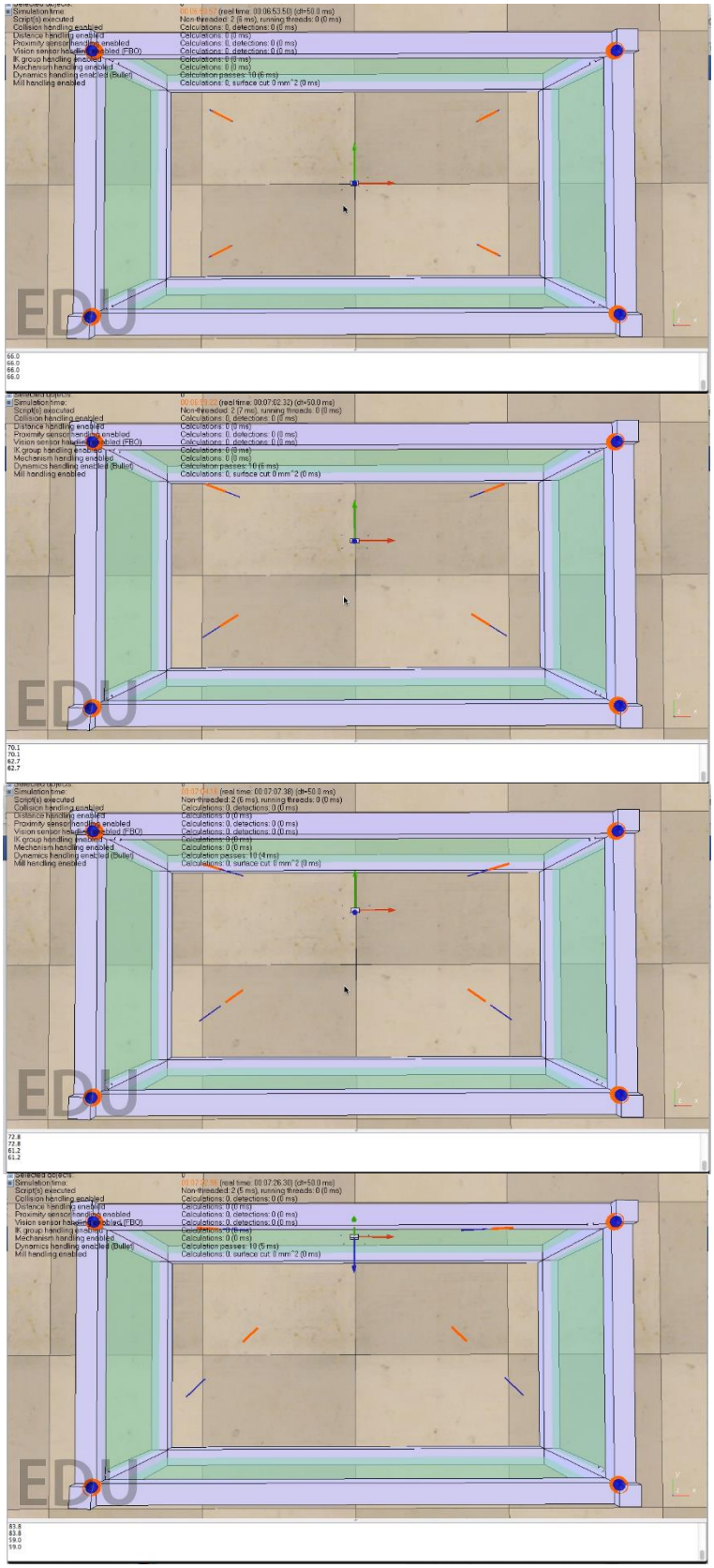

Figura 7.3- Sequência do movimento no eixo de Y

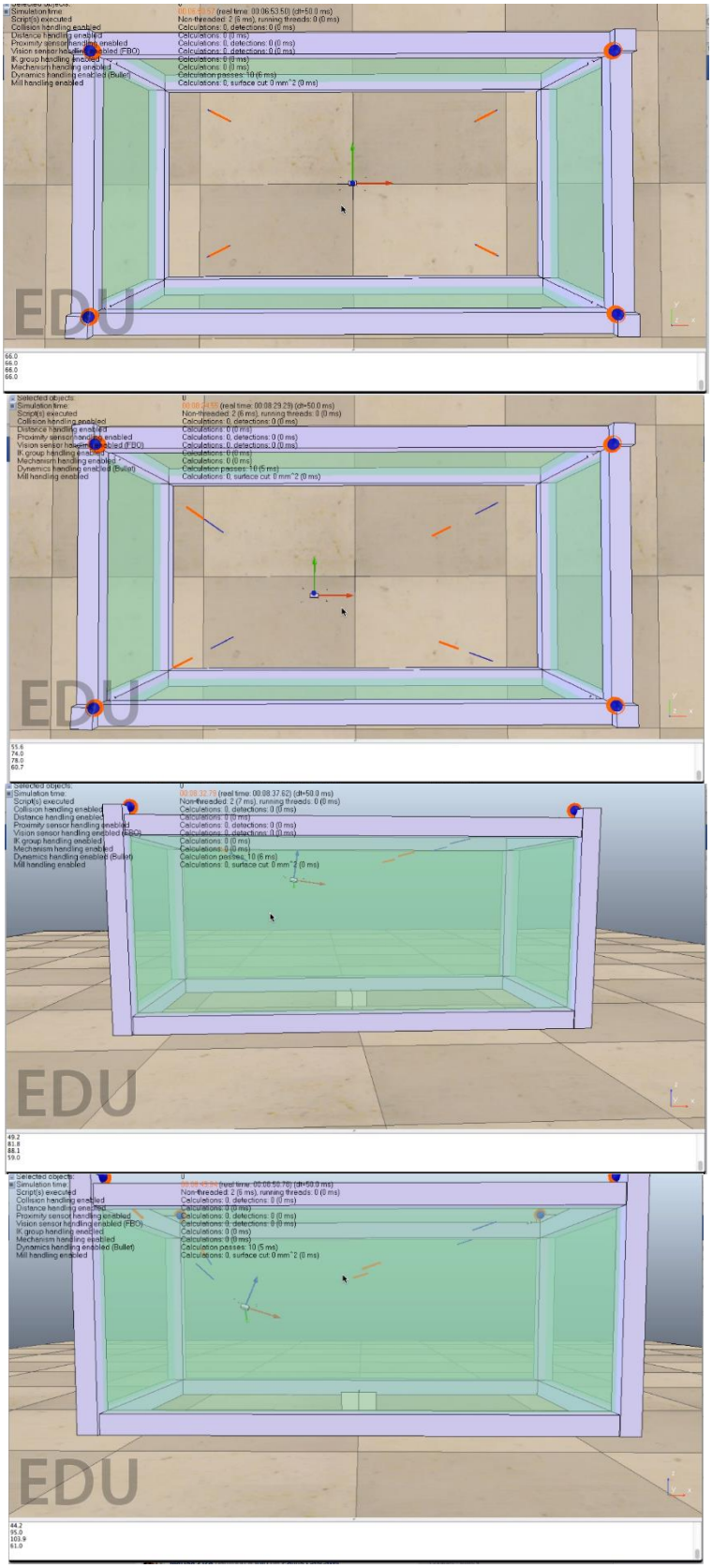

Figura 7.4- Sequência do quarto movimento de teste

Para a realização dos testes a aplicação de controlo teve de estar devidamente configurada para comunicar com o v-rep, começando por criar uma rede de trabalho com dois pc's caso a mesma ainda não exista e configurou-se a porta de comunicação na rede. Seguidamente introduziu-se a posição cartesiana (X, Y, Z) desejada no interface gráfico bem como o número de iterações desejado dando instrução para que sejam calculados os respetivos comprimentos dos cabos. O calculo é efectuado internamente na aplicação de controlo sem qualquer interferência do utilizador.

A aplicação de controlo transmite os valores de cada cabo ao v-rep iniciando a movimentação da plataforma. O movimento é lento e controlado, de forma a que não haja qualquer sobrecarga na tensão de nenhum dos cabos.

É possível verificar visualmente o movimento da plataforma bem como a resposta dinâmica dos cabos e dos restantes elementos do sistema. Em todos os casos é possível também consultar o comprimento respetivo de cada cabo na parte inferior de cada fração da sequência, refentes aos valores calculados na aplicação de controlo.

Nas Figuras 7.2, 7.3 e 7.4 é possível verificar que a plataforma não atinge por completo as coordenadas pretendidas e que existe uma pequena inclinação na plataforma, o que resulta da limitação ao nível de manter a tensão em todos os cabos, o que é normal neste tipo de robôs. À medida que a plataforma se afasta do seu ponto de origem a precisão diminui e a inclinação natural da plataforma aumenta. Quando são alteradas as coordenadas nos três eixos (x, y, z) simultaneamente os efeitos anteriormente descritos agravam-se

Os testes foram feitos com quinhentas iterações de forma a ser possível tirar ilustrações, o que na prática é um valor exagerado, a não ser em casos específicos como cargas muito frágeis.

8- Conclusões

Nesta dissertação é apresentado todo o percurso do desenvolvimento de um ambiente de simulação de um robô de cabos. O projeto obedeceu aos requisitos iniciais e adaptou-se a novos requisitos sem nunca se desviar do seu objetivo principal. Como referido anteriormente o *Contour Crafting* permite a construção de estruturas de elevadas dimensões, sendo o manuseamento, rapidez, precisão e segurança exemplos das características de maior valor. O desenvolvimento de simuladores é uma prática bastante útil em engenharia para conceção de protótipos. As dimensões reduzidas em relação ao produto final permitem, numa fase inicial, menores custos na produção desses protótipos. O projeto foi separado em várias fases, desde o estudo da arte, até ao controlo do simulador do robô.

O Estado da arte proporcionou a compreensão do problema proposto e das tecnologias a utilizar. Os exemplos estudados utilizam arquiteturas diferentes, e como consequência, têm características diferentes.

O modelo a utilizar foi escolhido segundo as características propostas, mas também tendo em conta os custos reduzidos na elaboração de um protótipo. A escolha recaiu então numa arquitetura de quatro cabos, sendo a única força descendente, aplicada em todos os momentos na plataforma, a força gravitacional. Este tipo de simulador é, à partida, o adequado para o objetivo pretendido. Com um baixo número de cabos (quatro), permite contenção de custos mas mantendo os três graus de liberdade pretendidos. Controlando o comprimento dos quatro cabos do robô, é possível movimentar a plataforma móvel. Uma vez que as reduzidas dimensões da plataforma permitem desprezar os ângulos *Yaw*, *Roll* e *Pitch*, esta desloca-se no espaço cartesiano mantendo sempre a direção paralela ao plano XY (base).

O volume de trabalho deste robô é semelhante aos demais robôs de cabos, em particular aos que só contêm cabos acima do centro de massa da ferramenta. O volume de trabalho aumenta da zona inferior para a superior, e é sempre inferior ao volume da estrutura do robô.

O interface desenvolvido para controlo do simulador permite uma maior facilidade e rapidez ao utilizador, não sendo preciso conhecimentos de diversas áreas, mas sim apenas ao nível de funcionamento básico de robôs.

A comunicação entre interface e simulador é configurável, podendo ser interna (um pc) ou externa (vários pc's) e permite a utilização deste simulador em vários sistemas operativos e através de uma rede *wi-fi* ou de cabos criada com essa finalidade.

O ambiente de simulação foi desenvolvido exatamente com a arquitetura e medidas pretendidas, o seu funcionamento é simples e totalmente controlável com o interface e visualmente apresenta as características possíveis permitidas pelo *software*.

Como trabalhos futuros podem ser sugeridos alguns trabalhos como: utilização dos graus de liberdade *Yaw*, *Roll* e *Pitch*; análise do volume de trabalho; consideração da elasticidade e peso dos cabos e da plataforma; consideração dos atuadores/motores responsáveis pela movimentação dos cabos; e principalmente a implementação de um protótipo real.

Referências

[1] B. Khoshnevis, Automated construction by countor crafting robotics and information technologies, Journal of Automation Construction — Special Issue: The Best of ISARC 2002 13 (1) (2004)

[2] B. Khoshnevis, R. Russel, H. Kwon, S. Bukkapatnam, Crafting large prototypes, IEEE Robotics & Automation Magazine (2001)

[3] www.nist.gov/el/isd/ms/robocrane.cfm

[4] www.nist.gov/el/isd/lunar-rover.cfm

[5] E. Ottaviano, M. Ceccarelli and M. De Ciantis: LARM: Laboratory of Robotics and Mechatronics, University Cassino, Cassino, Italy; A 4-4 Cable-Based Parallel Manipulator for an Application in Hospital Environment in: Proceedings of the 15th Mediterranean Conference on Control & Automation, July 27-29, 2007, Athens – Greece

[6] www.uni-due.de/imperia/md/content/mechatronik/aktuelles/dspace_200701_en.pdf

[7] www.europeanrobotics12.eu/news/wire-driven-parallel-robots.aspx

[8] Bruckmann, Tobias, Pott, Andreas (Eds.): Cable-Driven Parallel Robots, 2013

[9] www.eu-nited.net/robotics/press-room/cable-robot-ipanema-extended-version-for-engineers.html

[10] www.coppeliarobotics.com

[11] http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo

[12] www.cyberbotics.com/

[13] http://gazebosim.org/

[14] M. Martins: Arquitetura Robótica – Desenvolvimento de um

“cable robot” para construção, FEUP, 30 de Junho de 2014.

Anexos

Anexo A: Código da aplicação de controlo desenvolvida no Lazarus

I.T. para o produto da R9M (3ªParte).

```
unit Unit1;

{$mode objfpc}{$H+}

interface


uses

  Classes, SysUtils, FileUtil, lNetComponents, Forms, Controls, Graphics,

  Dialogs, StdCtrls, ExtCtrls, math;


const

platfromLength = 2;

platfromWidth =  1;

STARTX=0;

STARTY=0;

STARTZ=60;


type

  TPoint = record

    x,y,z: real;

  end;


  TCables = record
```

```
    L1,L2,L3,L4: real;
  end;

  { TRobot_Cable_Calculate }

  TRobot_Cable_Calculate = class(TForm)
    Button1: TButton;
    Button2: TButton;
    EditX: TEdit;
    EditY: TEdit;
    EditZ: TEdit;
    Image1: TImage;
    Label5: TLabel;
    Niteracoes: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    UDPSend: TLUDPComponent;
    Memo1: TMemo;
    Timer1: TTimer;
//    procedure Button1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
```

```
    procedure FormShow(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    function calcCabos(var ponto:TPoint):TCables;
  private
    { private declarations }
  public
    { public declarations }

    //pontoXYZ: TPoint;
    Cabos: array [1..4] of real;

    temp_pontoXYZ: TPoint;
    pontoXYZ: TPoint;
    A,B: array [1..4] of TPoint;
    CurrPosition: TPoint;
    pontofinal:TPoint;
     deltaPoint:TPoint;
     num_iter: integer;
     ITERATIONS: real;
  end;

var
  Robot_Cable_Calculate: TRobot_Cable_Calculate;
```

```
implementation

{$R *.lfm}

{ TRobot_Cable_Calculate }

function TRobot_Cable_Calculate.calcCabos(var ponto:TPoint):TCables;
var
  i: integer;
  compCabos:TCables;
begin
   A[1].X := -60; A[1].Y := -30; A[1].Z := 60;
  A[2].X := 60; A[2].Y := -30; A[2].Z := 60 ;
  A[3].X := 60; A[3].Y := 30; A[3].Z := 60;
  A[4].X := -60; A[4].Y := 30; A[4].Z := 60;

  pontoXYZ.x:=ponto.x;
  pontoXYZ.y:=ponto.y;
  pontoXYZ.z:=ponto.z;

  B[1].X := pontoXYZ.X - (platfromLength / 2);
  B[2].X := pontoXYZ.X + (platfromLength / 2);
  B[3].X := B[2].X;
  B[4].X := B[1].X;
```

```
B[1].Y := pontoXYZ.Y - (platfromWidth / 2);

B[2].Y := B[1].Y;

B[3].Y := pontoXYZ.Y + (platfromWidth / 2);

B[4].Y := B[3].Y;

B[1].Z := pontoXYZ.Z;

B[2].Z := pontoXYZ.Z;

B[3].Z := pontoXYZ.Z;

B[4].Z := pontoXYZ.Z;


//Calculate the Cabes Length and Enconders Counters

memo1.Clear;


 for i:=1 to 4 do

    begin

      Cabos[i] := ( (B[i].X-A[i].X)*(B[i].X-A[i].X) );

      Cabos[i] :=  Cabos[i] + ( (B[i].Y-A[i].Y)*(B[i].Y-A[i].Y) ) ;

      Cabos[i] :=  Cabos[i] + power(B[i].Z-A[i].Z,2);

      Cabos[i] := sqrt( Cabos[i]);

    end;


   compCabos.L1:=Cabos[1];

   compCabos.L2:=Cabos[2];

   compCabos.L3:=Cabos[3];

   compCabos.L4:=Cabos[4];
```

```
    calcCabos := compCabos;
end;

procedure TRobot_Cable_Calculate.Button2Click(Sender: TObject);

begin
  Button2.Enabled:=false;
 num_iter :=0;
 if strtofloat(EditX.Text) > 60 then EditX.Text :='60';
 if strtofloat(EditY.Text) > 30 then EditY.Text :='30';
 if strtofloat(EditZ.Text) > 60 then EditZ.Text :='60';
 if strtofloat(EditX.Text) < -60 then EditX.Text :='-60';
 if strtofloat(EditY.Text) < -30 then EditY.Text :='-30';
 if strtofloat(EditZ.Text) < 0 then EditZ.Text :='0';
 pontofinal.x:=strtofloat(EditX.Text);
 pontofinal.y:=strtofloat(EditY.Text);
 pontofinal.Z:=strtofloat(EditZ.Text);
 ITERATIONS:=strtofloat(Niteracoes.Text);

 temp_pontoXYZ.x:=CurrPosition.x;
 temp_pontoXYZ.y:=CurrPosition.y;
 temp_pontoXYZ.z:=CurrPosition.z;

  deltaPoint.x:=(pontofinal.x-CurrPosition.x)/ITERATIONS;
```

```
  deltaPoint.y:=(pontofinal.y-CurrPosition.y)/ITERATIONS;
  deltaPoint.z:=(pontofinal.z-CurrPosition.z)/ITERATIONS;

  Memo1.Clear;

  Timer1.Enabled:=true;
end;

procedure TRobot_Cable_Calculate.Button1Click(Sender: TObject);

begin
  Button2.Enabled:=false;
 num_iter :=0;

 pontofinal.x:=0;
 pontofinal.y:=0;
 pontofinal.Z:=60;
 ITERATIONS:=50;

 temp_pontoXYZ.x:=CurrPosition.x;
 temp_pontoXYZ.y:=CurrPosition.y;
 temp_pontoXYZ.z:=CurrPosition.z;

  deltaPoint.x:=(pontofinal.x-CurrPosition.x)/ITERATIONS;
```

```
  deltaPoint.y:=(pontofinal.y-CurrPosition.y)/ITERATIONS;
  deltaPoint.z:=(pontofinal.z-CurrPosition.z)/ITERATIONS;

  Memo1.Clear;

  Timer1.Enabled:=true;

end;



procedure TRobot_Cable_Calculate.FormShow(Sender: TObject);
begin
  CurrPosition.x:=STARTX;
  CurrPosition.y:=STARTY;
  CurrPosition.z:=STARTZ;
end;

procedure TRobot_Cable_Calculate.Timer1Timer(Sender: TObject);
var s: string;
    cabosresult: TCables;
begin
  inc(num_iter);
```

```
  temp_pontoXYZ.x:= temp_pontoXYZ.x + deltaPoint.x;
  temp_pontoXYZ.y:= temp_pontoXYZ.y + deltaPoint.y;
  temp_pontoXYZ.z:= temp_pontoXYZ.Z+ deltaPoint.z;
  cabosresult := calcCabos(temp_pontoXYZ);

  s:='';
   s := s + floattostrf(cabosresult.L1, ffFixed, 4,1) + ';' + floattostrf(cabosresult.L2,
ffFixed, 4,1) + ';' + floattostrf(cabosresult.L3, ffFixed, 4,1) + ';'+
floattostrf(cabosresult.L4, ffFixed, 4,1) + ';';
  memo1.Lines.Add(s);
  UDPSend.Connect('169.254.63.240', 8278);
UDPSend.SendMessage(s);

   if (num_iter = ITERATIONS ) then  begin
      CurrPosition.x := pontofinal.x;
      CurrPosition.y := pontofinal.y;
      CurrPosition.z := pontofinal.z;
      Timer1.Enabled:=false;
      Button2.Enabled:=True;
   end;
end;

end.
```

Anexo B: Código da aplicação em *software* V-rep

```
if (simGetScriptExecutionCount()==0) then
        ----------  LAZARUS ----------------
local socket = require("socket")
udp = assert(socket.udp())
udpRec = assert(socket.udp())
assert(udpRec:setsockname ('*', 8278))
udpRec:settimeout(0)
end
   ---------- V-REP -------------
simHandleChildScript(sim_handle_all_except_explicit)

local data, ip, port = udpRec:receivefrom()

if(data ~= nil) then
        function split(s, delimiter)
                result = {};
        for match in (s..delimiter):gmatch("(.-)"..delimiter) do
                table.insert(result, match);
        end
        return result;
end

vetor = split(data, ";");

cabo1=vetor[1];
cabo2=vetor[2];
cabo3=vetor[3];
cabo4=vetor[4];
```

```
simAddStatusbarMessage(cabo1)
simAddStatusbarMessage(cabo2)
simAddStatusbarMessage(cabo3)
simAddStatusbarMessage(cabo4)

        c1=(cabo1-(65.96))/100;
        c2=(cabo2-(65.96))/100;
        c3=(cabo3-(65.96))/100;
        c4=(cabo4-(65.96))/100;

joint1=simGetObjectHandle('L1')
        position=simSetJointTargetPosition(joint1,c1);

joint2=simGetObjectHandle('L2')
        position=simSetJointTargetPosition(joint2,-c2);

joint3=simGetObjectHandle('L3')
        position=simSetJointTargetPosition(joint3, -c3);

joint4=simGetObjectHandle('L4')
        position=simSetJointTargetPosition(joint4, c4);

end
```