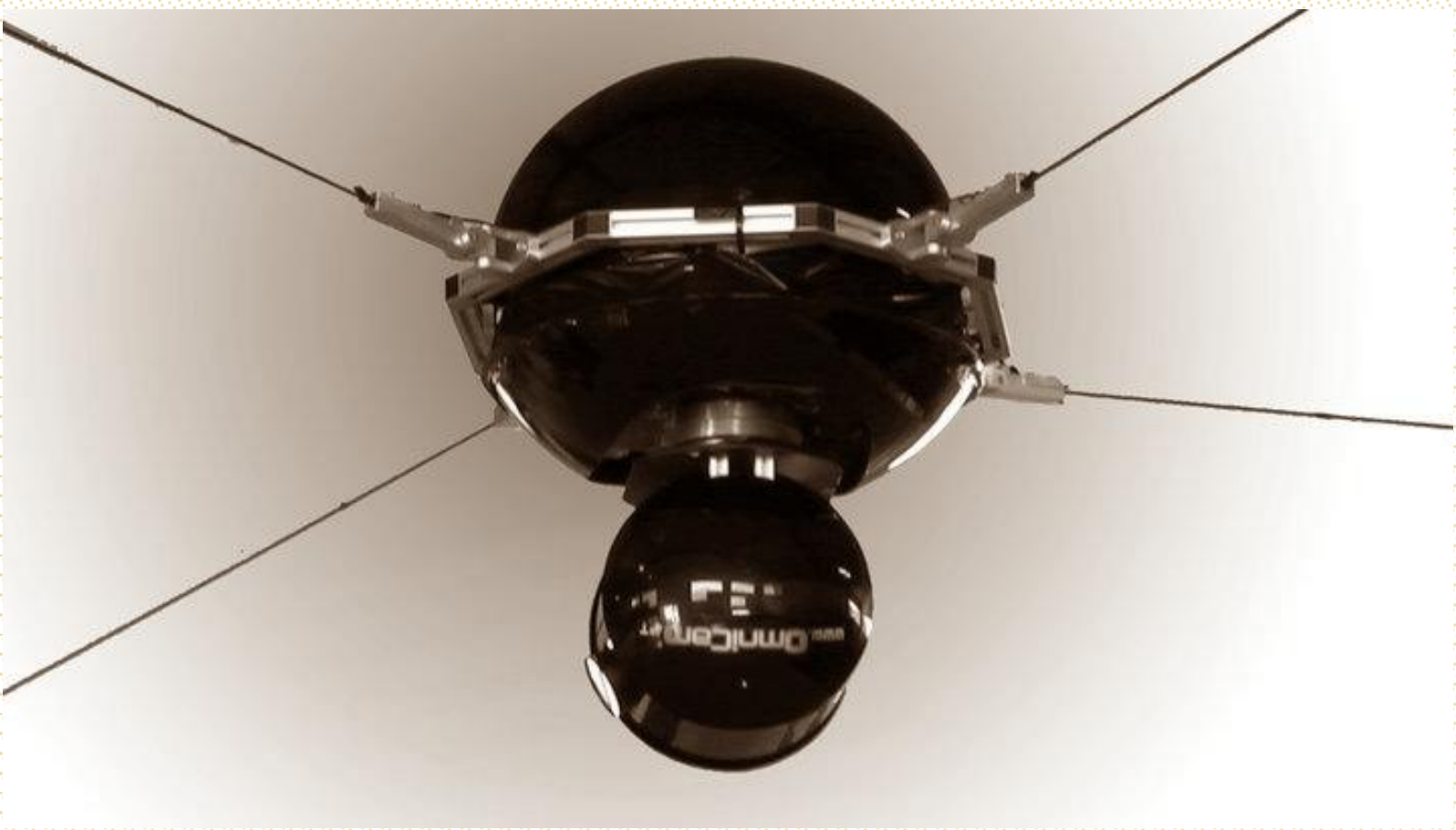


Rúben Daniel Mendes Pessoa Lopes

Robô paralelo conduzido por cabos para movimentação de objectos em espaços de trabalho muito amplos

Data: 24/01/2012

· U C ·

Universidade de Coimbra

FCTUC

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica de Coimbra

Robô paralelo conduzido por cabos para movimentação de objectos em espaços de trabalho muito amplos

Autor:

Rúben Daniel Mendes Pessoa Lopes

Júri:

Presidente: Professor Rui Paulo Pinto da Rocha
Orientador: Professor Lino José Forte Marques
Vogal: Professor Paulo Jorge Carvalho Menezes

Janeiro 2012

Resumo

Os robôs paralelos conduzidos por cabos são inspirados na plataforma de Stewart. Enquanto na plataforma de Stewart os actuadores são rígidos, neste caso são substituídos por cabos. Esta alteração permite aumentar drasticamente o espaço de trabalho.

Esta dissertação estuda métodos para implementar zonas de segurança num robô paralelo conduzido por cabos. Isto é, impedir que o robô se desloque para zonas com obstáculos ou permitir delimitar o espaço de trabalho. Para tal, é feito um estudo sobre robôs paralelos conduzidos por cabos. Analisa-se a cinemática directa e inversa deste tipo de robôs, dando maior atenção ao robô comandado por quatro cabos. É ainda feito um estudo da dinâmica destes sistemas.

Este trabalho teve como base o sistema OmniCam4Sky em desenvolvimento pela empresa OmniCam[1]. Este sistema consiste numa câmara de alta qualidade acoplada a um robô paralelo conduzido por cabos que permite posicionar a câmara numa qualquer posição dentro do seu espaço de trabalho.

A par com este estudo, criou-se um simulador em *Matlab* para testar os algoritmos a implementar e por fim, fez-se a implementação de algoritmos de movimentação da câmara, tendo em conta um conjunto de zonas de segurança num autómato *Movi-PLC* da Sew[2].

Palavras-Chave: Robô conduzido por cabo, Cinemática de robôs conduzidos por cabos, Segurança de Máquinas, Delimitação do espaço de trabalho.

[1] www.omnicam.pt
[2] http://www.sew-eurodrive.pt/

Abstract

Wire driven parallel robots are based on the Stewart platform. While on the Stewart platform the actuators are rigid, here a different approach is used. The rigid actuators were replaced by cables which allows an increase of the workspace.

The current dissertation, concentrates on the implementation of safety zones for a wire driven parallel robot. In other words, the aim is to stop the robot from traveling to areas populated by obstacles, or, outside of the working zone. In order to achieve this, a deep analysis of the robot was required, including robot kinematics and dynamics.

The approach used in this work is based on OmniCam4sky in development by OmniCam. This is a system that complies of a high quality camera mounted on the robot which allows position the camera in any position within the workspace.

A simulator was also developed in *Matlab*, to allow the study and reliability of new interfaces and visualization systems and finally it was developed the algorithms to move the camera, taking in account a set of safety zones on Sew Movi-PLC.

Key-Words: Cable suspended robots, Kinematics, Machine safety, Delimitation of the workspace.

Índice

Lista de Figuras

Acrónimos

CAN – Controller Area Network

DCOM – Distributed Component Object Model

DDE – Dynamic Data Exchange

DOF – Degrees Of Freedom

IPOS – Sistema de Controlo Sequencial Integrado

NIST – National Institute of Standards and Technology

OLE – Object Linking & Embeding

OPC – OLE for Process Control

PLC – Programmable Logic Controller

ST – Structured Text

1 – Introdução

Nos últimos anos, os robôs paralelos conduzidos por cabos têm vindo a ser utilizados em variadas aplicações. Este tipo de robôs tem origem na plataforma de Stewart [1], [2]. Os robôs paralelos são constituídos por uma base fixa e uma plataforma ligada à base através de actuadores. A plataforma Stewart (Figura 1.1) utiliza actuadores rígidos, por exemplo hidráulicos ou pneumáticos. Esta característica resulta num espaço de trabalho relativamente limitado. Uma das aplicações mais conhecida, baseada na plataforma de Stewart são os simuladores de voo (Figura 1.2).

Figura 1.1: Plataforma de Stewart [36]

Nos robôs paralelos conduzidos por cabos os actuadores rígidos, usados na plataforma de Stewart, são substituídos por cabos. O sistema é constituído por múltiplos motores, usados para enrolar e desenrolar os cabos que estão ligados a uma carga. A posição da carga é alterada enrolando ou desenrolando os cabos de cada motor simultaneamente, evitando sistematicamente que algum dos cabos fique com folga. É de notar que ao termos cabos como actuadores os movimentos ficam limitados, pode-se puxar a carga mas não se pode empurrar. Esta característica requer modelos dinâmicos que satisfaçam a condição de tensões positivas nos cabos, o que implica que os métodos tradicionais na análise de robôs não possam ser aplicados. É necessário ter em conta o erro em posição devido a elasticidade dos cabos.

Este tipo de robôs permitem colmatar a limitação ao nível de espaço de trabalho da plataforma de Stewart, aumentando-o drasticamente o e permitindo alargar o número de aplicações baseadas no princípio de manipuladores paralelos. Outra grande vantagem destes sistemas está na sua velocidade, o facto das partes móveis serem extremamente leves permite níveis de aceleração e velocidade elevados na deslocação da carga. Esta característica ainda resulta numa boa eficiência energética e torna o sistema adequado

para movimentar cargas pesadas. A simplicidade da estrutura dos robôs paralelos conduzidos por cabos permite que o sistema seja relativamente fácil de transportar e fácil de reconfigurar.

Figura 1.2: Simulador de Voo [37]

Resumindo, as vantagens dos robôs paralelos conduzidos por cabos são:

- Grande espaço de trabalho;
- Elevada aceleração;
- Elevada velocidade;
- Eficiência energética;
- Adequado para movimentar cargas pesadas;
- Facilidade em transportar o sistema;
- Facilidade de reconfigurar.

Os robôs paralelos conduzidos por cabos podem ser divididos em dois grupos, os "fully-constrained" (totalmente restritos) e os "under-constrained" (parcialmente controlados) [6]. No primeiro caso, a posição da carga pode ser determinada em função do comprimento dos cabos (Figura 1.3b). No segundo caso, a posição da carga não pode ser completamente determinada tendo apenas em conta o comprimento dos cabos, tem de estar presente a acção da gravidade (Figura 1.3a). É condição necessária, mas não

suficiente para que uma plataforma destas tenha *n* graus de liberdade (DOFs), ser conduzida por um mínimo de $m = n + 1$ cabos [3]. O grupo dos parcialmente controlados é também chamado de manipuladores de cabos suspensos, porque a estrutura é constituída por uma carga suspensa.

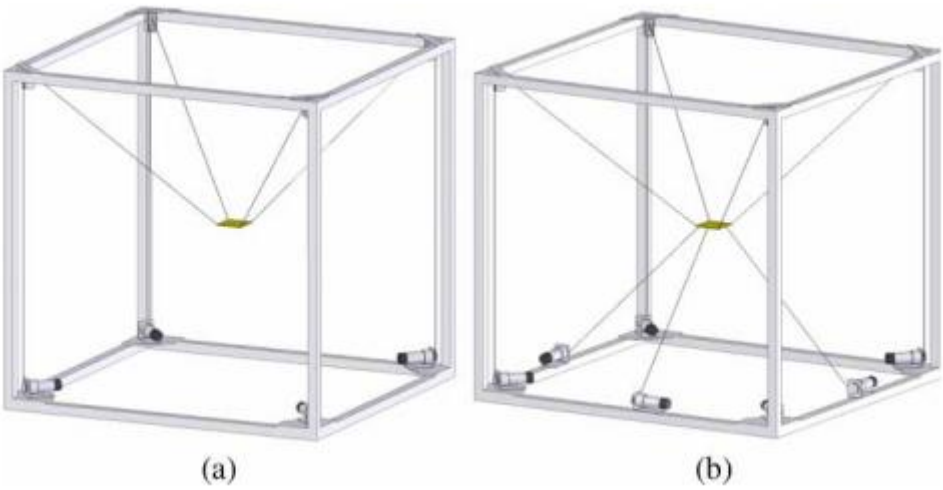


Figura 1.3: Tipos de robôs paralelos conduzidos por cabos: (a) parcialmente restrito e (b) totalmente restrito [6]

Entre as várias aplicações onde se podem utilizar os robôs paralelos conduzidos por cabos podem-se referir: manipulação de cargas pesadas, inspecção e reparação em estaleiros e hangares de aviação, robôs de resgate, limpeza de áreas de desastre e interacção com locais perigosos. Um outro exemplo onde é usado este tipo de robôs é no posicionamento de câmaras de vídeo.

Utilizar uma câmara de vídeo como carga trouxe uma grande inovação no mundo do desporto. Hoje em dia é possível estar a ver um jogo de futebol como se estivéssemos a jogar um jogo de vídeo. Estes sistemas permitem recolher imagens a qualquer ponto do campo, a alturas que podem variar desde distâncias próximas ao chão (1 a 2 metros) até à altura do estádio. Obviamente que a utilização destes sistemas não se restringe ao futebol, podendo ser utilizados em qualquer tipo de evento que ocorra em espaços de elevada dimensão. Uma das principais vantagens destes sistemas é a possibilidade de se movimentar por cima dos intervenientes (espectadores, jogadores, artistas, etc.). Esta capacidade, embora muito útil, acarreta riscos elevadíssimos que é necessário acautelar, tornando os sistemas de segurança muito importantes. O sistema tem de garantir que a câmara não colide com os intervenientes ou possíveis obstáculos.

Um exemplo deste tipo de aplicação é o sistema de movimentação tridimensional de câmara OmniCam4Sky (Figura 1.4). O sistema tem quatro motores acoplados a tambores onde são enrolados os cabos que sustentam a câmara. O espaço de trabalho da câmara é definido pela posição dos tambores. Para ajudar a estabilizar a câmara, a estrutura que a suporta tem um giroscópio acoplado. O posicionamento da câmara é controlado enrolando e desenrolando os cabos. O comando é feito por dois joysticks e processado num *Movi-PLC*, enviando os comandos para os controladores

por fibra óptica. Dois dos cabos que sustentam a câmara contêm fibra óptica no interior para transmitir os dados de controlo da câmara e a imagem em alta definição.

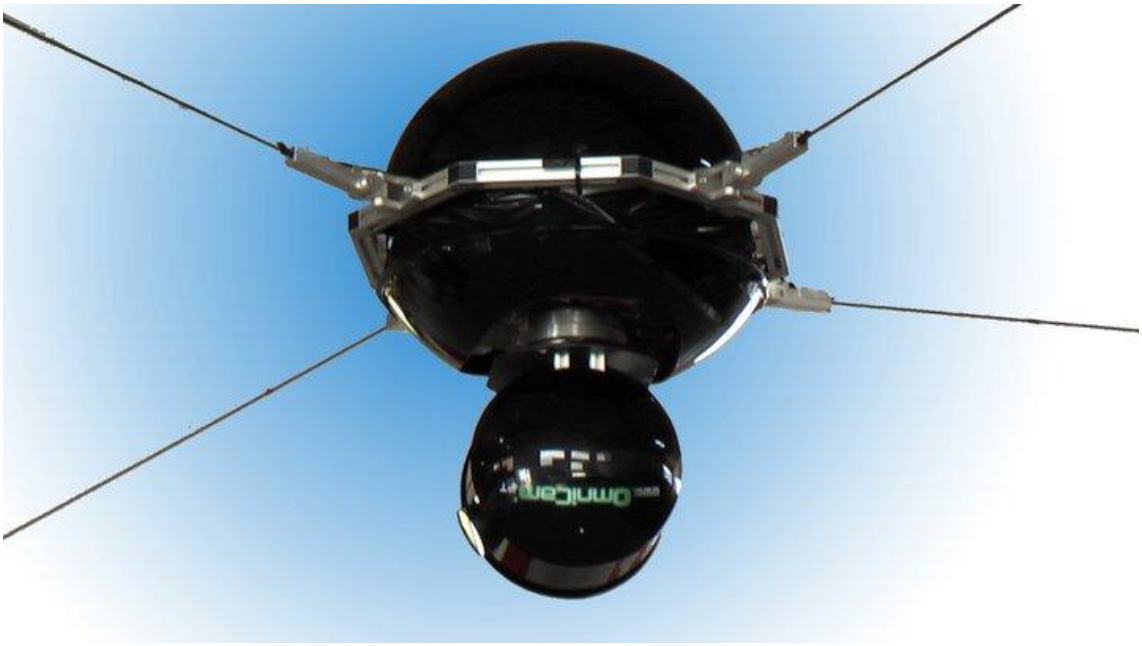


Figura 1.4: OmniCam4Sky

1.1 – Objectivos

Esta dissertação tem como objectivo o estudo de Robôs paralelos conduzidos por cabos para movimentação de objectos em espaços de trabalho muito amplos, e melhorar as condições de segurança dum sistema já existente.

Resumindo, os principais objectivos são:

- Análise cinemática de sistemas paralelos conduzidos por cabos;
- Determinar as equações que descrevem a dinâmica do sistema;
- Melhorar a segurança do sistema já existente com a implementação dum algoritmo que evita obstáculos;
- Optimizar a área de trabalho, isto é, permitir que a carga tenha acesso a todo o espaço de trabalho disponível;
- Estudar a comunicação OPC (OLE for Process Control) e a sua possível utilização no presente sistema.

1.2 – Estrutura

Neste capítulo introdutório é feita uma pequena apresentação aos robôs paralelos, dando mais ênfase aos robôs paralelos conduzidos por cabos, e é explicado o funcionamento geral do sistema que se propõe melhorar.

No capítulo 2 é feito o levantamento do estado da arte deste tipo de sistemas, referindo estudos e aplicações feitas para robôs paralelos conduzidos por cabos. Também é apresentada a estrutura do sistema OmniCam4Sky referindo os dispositivos mais importantes utilizados no sistema com uma breve descrição das principais características dos mesmos.

No capítulo 3 são apresentados os cálculos da cinemática e dinâmica de sistemas conduzidos por cabos. Descreve-se o caso particular do sistema em estudo mas também é feita uma abordagem a nível mais geral, permitindo alargar o cálculo para diferentes configurações.

No capítulo 4 é feita a descrição do simulador construído em *Matlab* para poder testar os algoritmos a implementar e é feita a descrição dos algoritmos implementados no sistema.

No capítulo 5 começa-se por fazer uma breve descrição da comunicação OPC (OLE for Process Control) referindo as suas vantagens e é explorada a possibilidade de usar o OPC para criar um interface para o sistema.

No capítulo 6 são apresentados os testes e resultados experimentais, bem como as conclusões do trabalho realizado e sugestões para trabalho futuro.

2 – Robôs paralelos actuados por cabos

2.1 – Todos

Os robôs paralelos conduzidos por cabos são um caso particular dos robôs paralelos baseados na plataforma de Stewart. Existem muitos estudos para robôs baseados na plataforma de Stewart. No entanto, apesar das semelhanças entre estes e os robôs paralelos conduzidos por cabos, os estudos feitos não podem ser directamente aplicados devido às particularidades dos cabos. Os cabos só podem exercer força numa direcção, isto é, os cabos podem puxar uma carga mas não a podem empurrar. Nos últimos tempos, os estudos feitos em robôs paralelos conduzidos por cabos têm aumentado devido à sua velocidade de deslocação, a grandes espaços de trabalho e à facilidade de transporte. Um dos primeiros manipuladores paralelos conduzidos por cabos foi o Robocrane do NIST (National Institute of Standards and Technology). Este é um robô baseado na plataforma de Stewart invertida (Figura 2.1).

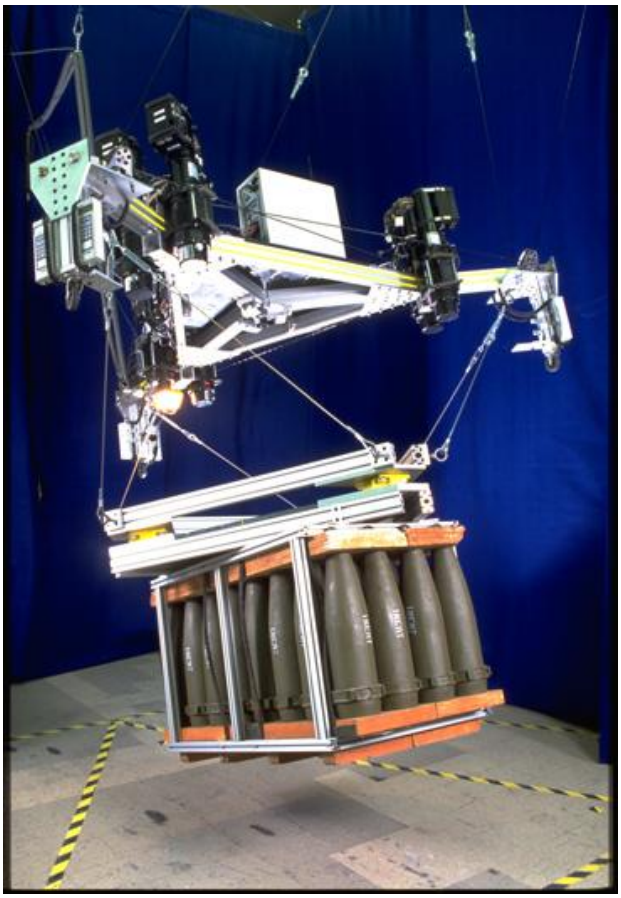


Figura 2.1: Protótipo do Robocrane [NIST].

Como os cabos só podem aplicar forças numa direcção, é necessário redundância para desenhar um robô conduzido por cabos totalmente controlado. Vários investigadores, entre os quais Kawamura, Ito e Kino [25] [26], têm estudado a controlabilidade de robôs conduzidos por cabos, tendo concluído que para um robô com *n* graus de liberdade, são necessários *n+1* cabos para manter tensões positivas em todos os cabos. Isto não se aplica a manipuladores que utilizam a gravidade para manter as tensões dos cabos positivas. O total de configurações possíveis com as tensões dos cabos positivas é o espaço de trabalho do manipulador.

Existem vários estudos de estática e cinemática de plataformas Stewart e alguns estudos sobre estática e cinemática em sistemas suspensos por cabos. Roberts, Graham e Lippitt [27] apresentaram uma solução para verificar se uma posição e orientação é estaticamente possível para um sistema conduzido por cabos com redundância. Esta solução é muito útil para encontrar o espaço de trabalho onde um robô suspenso por cabos esteja em equilíbrio estático ("espaço de trabalho estaticamente alcançável").

O método para calcular uma solução discreta para o espaço de trabalho utilizado por Roberts et al. é baseado na utilização da matriz jacobiana [27]. Neste método é utilizada análise geométrica do espaço nulo da matriz jacobiana para verificar se uma posição do manipulador é possível de alcançar mantendo a tensão nos cabos positiva. Roberts et al. fizeram também um pequeno estudo sobre tolerância a falhas no caso de se partir um cabo para robôs paralelos conduzidos por quatro cabos.

Foram feitos vários estudos, para além do mencionado anteriormente, sobre espaço de trabalho dos robôs paralelos conduzidos por cabos. Outro método usado para encontrar o espaço de trabalho deste tipo de sistemas têm por base a formulação da teoria do invólucro convexo (convex hull). Sendo o "convex hull" o menor poliedro convexo que pode ser formado em torno de um conjunto de pontos. Stump e Kumar [28] apresentaram também outro método analítico para calcular as fronteiras do espaço de trabalho, usando as desigualdades derivadas de Farkas' Lemma [29].

Os modelos analíticos apresentados anteriormente não têm em consideração a acção de forças externas como é o caso da gravidade. Pelo contrário, os modelos discretos consideram as forças externas. Normalmente, as tensões máximas e mínimas dos cabos são tidas em conta por métodos de restrição para o espaço de trabalho.

Normalmente, os robôs conduzidos por cabos são projectados para evitar as colisões entre os cabos. Contudo, Wischnitzer et al. [35] estudaram o espaço de trabalho para robôs paralelos conduzidos por cabos permitindo que os cabos colidam entre eles para aumentar o espaço de trabalho. As experiências feitas, permitindo colisões entre os cabos, demonstraram que o espaço de trabalho é aumentado e que o modelo, sem considerar fricção entre os cabos, é suficiente para aplicações reais com um erro no tamanho dos cabos na ordem de 1%.

Os estudos descritos anteriormente não tiveram em conta a elasticidade dos cabos. Koratem e Bamdad [34] estudaram espaços de trabalho de robôs paralelos com dois e três cabos tendo em conta a sua elasticidade. Os resultados obtidos mostraram que para algumas configurações de robôs paralelos conduzidos por cabos a elasticidade dos cabos não deve ser ignorada.

Outra causa de erros nos robôs paralelos conduzidos por cabos deve-se à sobreposição dos cabos quando são enrolados nas bobines. Esta sobreposição altera o diâmetro das bobines. Quando esta alteração não é tida em conta, acrescem erros ao sistema. Uma forma utilizada para ultrapassar este problema é colocar um actuador linear com roldanas nas bobines que enrolam os cabos.

Existem diferentes abordagens para obter as equações da dinâmica de uma plataforma paralela conduzida por cabos paralelos. As mais utilizadas são:

- Método de Newton-Euler;
- Método de Lagrange;
- Princípio do trabalho virtual.

Num sistema suspenso por cabos, a inércia dos cabos pode ser considerada insignificante. A aplicação das fórmulas de Newton-Euler é fácil e eficiente para este tipo de sistemas.

Existem configurações nos manipuladores paralelos que levam à perda de graus de liberdade. Estas configurações são chamadas de singularidades. Os manipuladores paralelos podem ter várias singularidades no seu espaço de trabalho. A formulação e análise de singularidades dos robôs paralelos conduzidos por cabos são idênticas aos robôs com actuadores rígidos. Quando se faz o estudo do espaço de trabalho para um robô paralelo conduzido por cabos é importante verificar quais as configurações que levam a singularidades.

A rigidez (stiffness) é um factor a ter em conta nos robôs paralelos, pois determina os desvios sofridos na carga. O aumento da rigidez dos robôs paralelos conduzidos por cabos reduz a vibração nos cabos. É uma ajuda para determinar os erros de posicionamento da carga e é importante no controlo do sistema. A rigidez do sistema pode ser aumentada escolhendo o posicionamento dos cabos. O primeiro estudo de rigidez em robôs paralelos conduzidos por cabos foi feito por Dagalakis et al. na plataforma Robocrane do NIST.

2.2 – OmniCam4Sky

Visto que os algoritmos para limitação do espaço de trabalho foram implementados num sistema já existente faz todo o sentido começar por fazer uma apresentação do sistema utilizado. Este capítulo apresenta a arquitectura global do sistema, descrevendo os dispositivos e software utilizados.

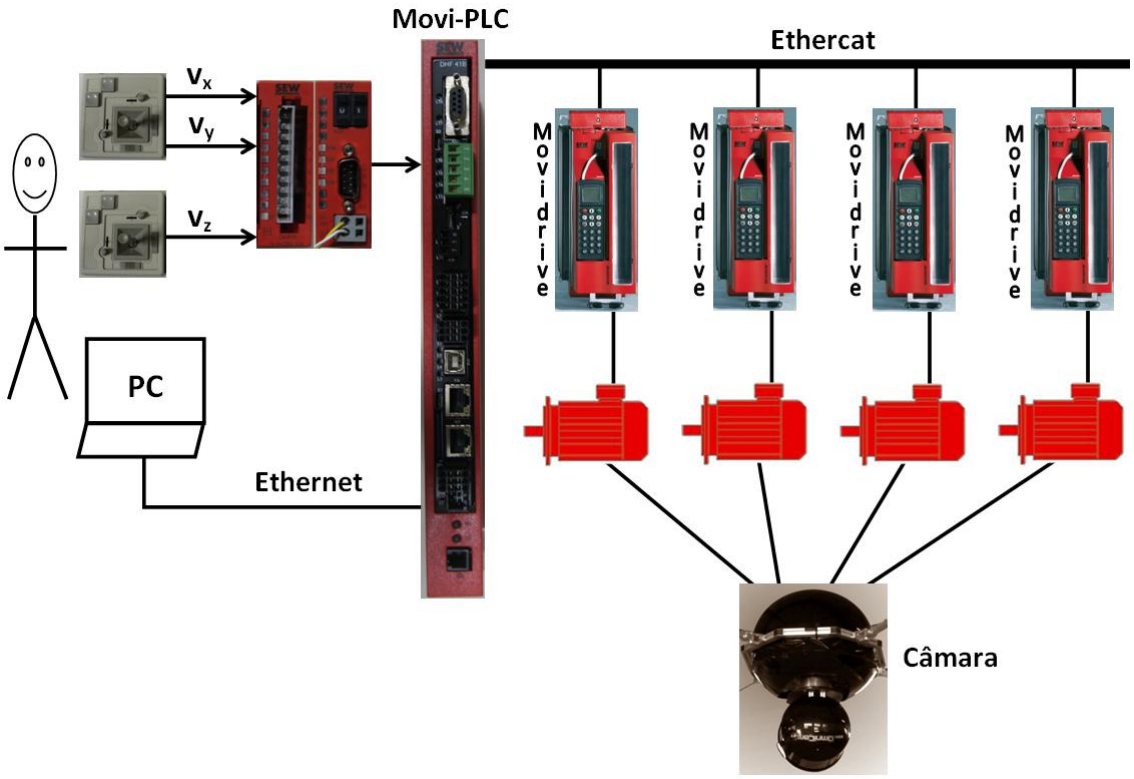


Figura 2.2: Arquitectura base do sistema OmniCam4Sky

A Figura 2.2 representa o esquema global do sistema OmniCam4Sky. É de notar que o esquema representado é um modelo simplificado, na realidade existe outro sistema acoplado a funcionar em simultâneo que actua no caso de existir alguma falha.

Podemos ver pela Figura 2.2 que o *Movi-PLC* tem como entrada dois joysticks, um de dois eixos e outro de um eixo. Os joysticks são ligados à expansão de entradas analógicas, que por sua vez está ligada a um acoplador de rede CAN que faz ligação com o *Movi-PLC*. Estes joysticks permitem comandar a posição *(x,y,z)* da câmara. O computador ligado por Ethernet ao *PLC* serve de interface com o utilizador para parametrização inicial do sistema e visualização durante operação. Esta interface torna-se muito útil na medida em que o utilizador nem sempre tem uma boa perspectiva da posição da câmara devido as dimensões do espaço de trabalho. O sistema de visualização permite ter noção da posição da câmara, ultrapassando esta dificuldade.

É possível ainda verificar que o sistema tem quatro inversores *Movidrive* ligados ao *PLC*. Estes inversores já têm incorporado algumas funções de controlo permitindo tirar alguma carga computacional ao *Movi-PLC*. Por sua vez, cada um dos inversores está ligado a um motor que controla o enrolar e desenrolar dos tambores. Sendo que, este enrolar e desenrolar dos tambores posiciona a câmara. É de notar que, apesar de não ser objecto de estudo desta dissertação, a câmara pode ser direccionada permitindo alterar a sua inclinação.

A comunicação entre o *Movi-PLC* e os *Movidrive* (Figura 2.2) é feita através de um canal Ethercat com um refrescamento na ordem dos 5 ms enquanto a comunicação entre o *Movi-PLC* e o computador é feita por OPC com um refrescamento na ordem dos 100 ms. Enquanto na comunicação com os *Movidrive* é necessário um refrescamento em tempo real para conduzir a câmara, na comunicação com o computador não é necessário estar a sobrecarregar o sistema com um refrescamento muito elevado visto que é para efeitos de visualização. De notar que os motores que enrolam e desenrolam os cabos nos tambores têm acoplado um encoder para permitir o controlo em posição dos mesmos. Os cabos que sustentam a câmara são feitos de Kevlar, uma fibra sintética de aramida usada também em coletes à prova de bala, e cada um dos cabos pode suportar até 272 Kg.

2.2.1 – Inversor Movidrive

São usados quatro controladores *Movidrive MDX61B* (Figura 2.3) no sistema para fazer o controlo dos quatro motores que enrolam e desenrolam os cabos nos tambores. O *Movidrive MDX60/61B* é um variador tecnológico de alto desempenho para accionamentos dinâmicos na gama de potências entre 0.55 e 315 kW, que permite aplicações variadas devido a amplas opções de expansão. Estes controladores já trazem um vasto número de funções de série facilitando a sua utilização. As principais características destes controladores são: extensa gama de potências, elevada capacidade de sobrecarga, um conceito modular, posicionamento e sistema de controlo sequencial integrados IPOS.

Figura 2.3: Movidrive

O posicionamento e sistema de controlo integrados possibilitam tirar alguma carga computacional ao nível superior (no caso em estudo é ao *Movi-PLC*), podendo em alguns casos substituí-lo.

Podemos enumerar algumas das características do sistema de controlo sequencial IPOS:

- Com encoder de realimentação, oferece um posicionamento ponto a ponto de alto desempenho.
- Execução do programa independentemente do encoder de realimentação e do modo de operação.
- Pode executar vários programas do utilizador/tarefas paralelamente e independentemente uma da outra.
- Possibilidade de acesso a opções existentes:
  - Carta de expansão de E/S
  - Interfaces de bus de campo
  - Carta de operação síncrona
- Amplas opções de comunicação.
- Processamento de sinais de entrada/saída binários e analógicos.
- Posicionamento com velocidade de deslocação e rampa de posicionamento seleccionável.
- Processamento dos sinais do encoder absoluto.
- Funções de monitorização e de estado:
  - Monitorização do erro de atraso
  - Indicação da posição
  - Fins de curso de software e de hardware
  - Monitorização do encoder
- São possíveis alterações das seguintes funções durante o deslocamento:
  - Posição destino
  - Velocidade de deslocação
  - Rampa de posicionamento
  - Binário

2.2.2 – O controlador de movimentos

Movi-PLC

O *Movi-PLC* (Figura 2.4) é um controlador para conversores de frequência com capacidade para ser programado em qualquer uma das cinco linguagens de programação definidas pela norma IEC 61131-3 [31], permitindo criar soluções simples e eficientes de accionamento, processamento lógico e controlo sequencial.

No caso da plataforma "Basic" do *Movi-PLC* é possível executar movimentos coordenados de eixos individuais, integrar entradas e saídas externas e interfaces Homem Máquina para accionamentos.

A plataforma "Advanced" do *Movi-PLC* têm todas as características referidas para a plataforma "Basic" e caracteriza-se ainda pela maior variedade de interfaces bem como um elevado desempenho que possibilita cálculos complexos, como é o caso de movimentos interpolados (usado do sistema implementado). Uma das características importantes de referir da plataforma "Advanced" é que permite uma conexão directa ao nível de controlo através da interface Ethernet integrada.

O controlador *Movi-PLC* "Advanced" dispõe de várias interfaces de comunicação. É de referir as interfaces RS485 que podem conectar, por exemplo uma interface Homem Máquina. Podemos enumerar duas interfaces de system bus CAN que tem como principal objectivo a conexão de vários conversores e a integração de módulos de expansão de entradas e saídas. Uma das interface que permite comunicar com um computador para programar o *Movi-PLC* é a interface de comunicação Ethernet. Este interface também permite criar um sistema de comunicação baseado em OPC (OLE for Process Control).

Figura 2.4: Exemplos de controladores Movi-PLC

A Configuração, parametrização e programação do *Movi-PLC* é feita através do software *Movitools MotionStudio*. A conexão entre o controlador *Movi-PLC* e o computador é feita através da interface de comunicação via Ethernet.

Expansão de Entrada/Saída para Movi-PLC

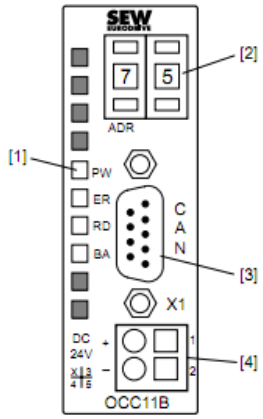
O sistema de Entradas/Saidas *Movi-PLC* modular amplia as interfaces digitais e analógicas do controlador *Movi-PLC*. Um sistema de Entradas/Saídas para o *Movi-PLC* inclui um acoplador de rede no qual é possível conectar um máximo de 32 módulos de Entrada/Saída através de um barramento situado na parte traseira. A comunicação com o *Movi-PLC* é feita através do system bus. O sistema permite conectar até 126 acopladores de rede.

Vantagens do sistema Entradas/Saídas *Movi-PLC*:

- Conexão de alto desempenho no controlador *Movi-PLC* através do SBus da unidade modular de máquina;
- Integração optimizada no software de programação *PLC editor* do *Movitools MotionStudio*;
- Diversas opções de combinação, permitindo a implementação de soluções flexíveis e personalizadas.

Acoplador de rede OCC11B

O acoplador de rede CAN OCC11BB (Figura 2.5) conecta o sistema *I/O Movi-PLC* com o controlador *Movi-PLC* através do system bus da *Sew-Eurodrive*. Este acoplador suporta todas as taxas de transmissão CAN.

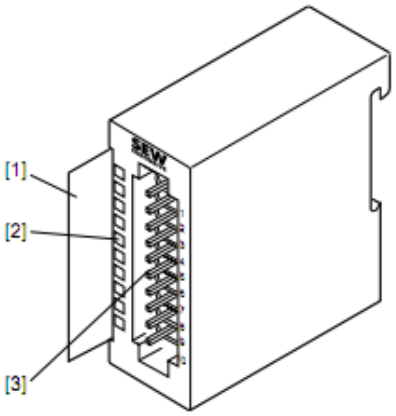


[1] Indicadores de estado LED
[2] Seletor de endereço para ajustar taxa de transmissão e módulo ID
[3] Conector de rede CAN
[4] Conexão para tensão de alimentação de 24 V_{CC} externa

Figura 2.5: Acoplador de rede OOC11B

Módulo de entradas analógicas OAI41B

O módulo de entradas analógicas OAI41B (Figura 2.6) tem quatro entradas, cujas funções podem ser parametrizadas individualmente ou desactivadas. O módulo está preparado para entradas entre os -10 e +10 Volts. A expansão analógica é utilizada para ligar os joysticks que permitem ao utilizador conduzir a câmara no seu espaço de trabalho. Este é ligado ao acoplador de rede descrito anteriormente.



[1] Etiqueta para o endereço do bit com descrição
[2] Indicador de estado LED
[3] Régua de conectores

Figura 2.6: Módulo de entradas analógicas OAI41B

2.2.3 – Ambiente de programação

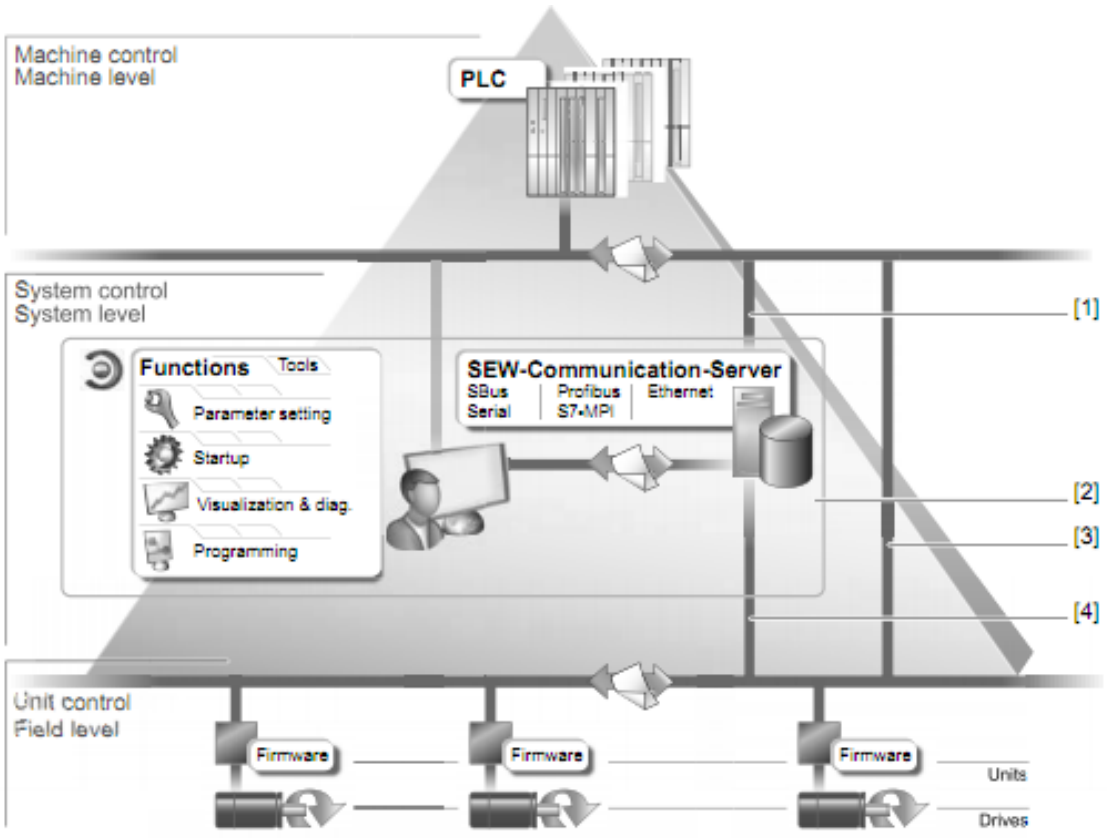
MotionStudio

Movitools MotionStudio é um software da *Sew-Eurodrive* que permite trabalhar com diversos dispositivos. Este software dispõe de inúmeras aplicações de alto desempenho para colocar em operação e para diagnóstico de todas as unidades de *Sew-Eurodrive*.

O pacote de software permite executar as seguintes tarefas com consistência:

- Estabelecer comunicação com as unidades;
- Executar funções com as unidades.

A Figura 2.7 ilustra as principais funções do *Movitools MotionStudio* num sistema.



[1] Communication channel for fieldbus or Industrial Ethernet
[2] MOVITOOLS® MotionStudio software package with integrated SEW Communication Server
[3] Communication between fieldbus stations or Industrial Ethernet stations
[4] Communication channel to SBus (CAN) or serial via interface adapter

Figura 2.7: Comunicação MotionStudio

O pacote de software *Movitools MotionStudio* já tem integrado um servidor de comunicação para estabelecer a comunicação com as unidades. Este servidor permite criar canais de comunicação. Os canais de comunicação possibilitam às unidades comunicar usando as suas opções de comunicação. É possível trabalhar com quatro canais de comunicação em simultâneo.

O software suporta os seguintes tipos de comunicação:

- Serial (RS-485);
- Ethernet;
- EtherCAT;
- Fieldbus (PROFIBUS DP/DP-V1).

3 – Modelação

3.1 – Cinemática

3.1.1 – Sistema em duas dimensões

Para poder perceber o conceito dos robôs conduzidos por cabos começou-se por modelar o sistema em duas dimensões. Para modelar o sistema é importante determinar a cinemática directa e indirecta do sistema. Na Figura 3.1, podemos ver o esquema de uma configuração em duas dimensões. O comprimento dos cabos é representado por *L1* e *L2*. Por simplicidade vamos considerar que ambos os cabos são fixos à mesma altura *h*.

Cinemática directa

Observando a Figura 3.1, e tendo em conta o teorema de Pitágoras obtêm-se:

$$\begin{cases} y = h - d \\ x = \sqrt{{L_1}^2 - d^2} \end{cases} \tag{3.1}$$

Onde *x* e *y* são as coordenadas da carga, e *d* é a distância vertical da carga ao ponto de suporte dos cabos. Para trabalhar a equação anterior de forma a ter as coordenadas da carga em função dos comprimentos dos cabos vamos ter em conta o esquema da Figura 3.2.

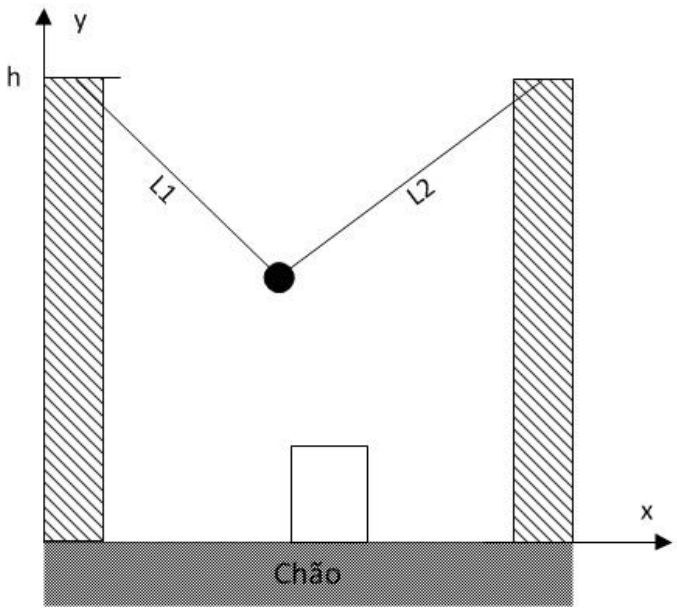


Figura 3.1: Sistema em duas dimensões

Os dois cabos ligados à carga formam um triângulo que pode ser dividido em dois triângulos rectos. É de notar que as forças que actuam na carga são, as tensões dos cabos e a gravidade. Num caso extremo a distância *d* pode ser nula, não é possível ter a carga acima do ponto onde são fixos os cabos.

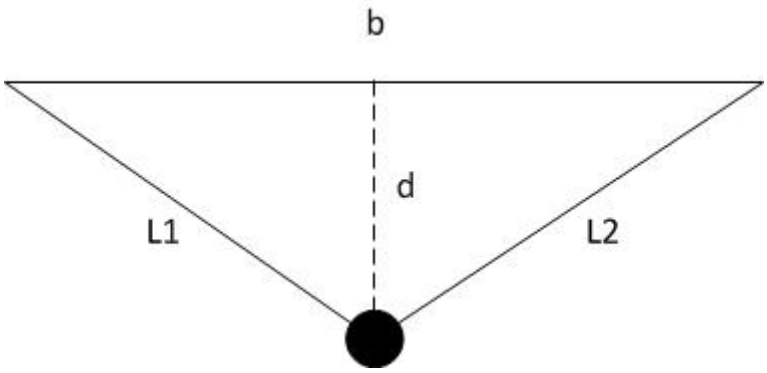


Figura 3.2: Esquema geométrico do sistema 2D

O teorema de Heron [32] diz que,

$$A = \sqrt{s(s - L_1)(s - b)(s - L_2)} \tag{3.2}$$

Onde *s* é metade do perímetro do triângulo, isto é,

$$s = \frac{L_1 + b + L_2}{2} \tag{3.3}$$

Sabendo que a área do triângulo é dada pela equação,

$$A = \frac{1}{2} b.d \tag{3.4}$$

Igualando as duas equações e resolvendo em ordem a *d*, obtemos:

$$d = \frac{\sqrt{s(s - L_1)(s - b)(s - L_2)}}{\frac{1}{2} b} \tag{3.5}$$

Substituindo a equação (3.5) na equação (3.1) obtêm-se as coordenadas *(x,y)* da carga em função dos comprimentos dos cabos:

$$\begin{cases} y = h - 2\dfrac{\sqrt{s(s-L_1)(s-b)(s-L_2)}}{b} \\ x = \sqrt{{L_1}^2 - \dfrac{4\sqrt{s(s-L_1)(s-b)(s-L_2)}}{b^2}} \end{cases} \tag{3.6}$$

A equação (3.6) dá-nos a cinemática directa do sistema em duas dimensões da Figura 3.1.

Cinemática inversa

Se tivermos em conta que os comprimentos de *L1* e *L2* são as distâncias da carga às posições onde cada um dos cabos é fixo, facilmente obtemos a cinemática inversa.

$$\begin{cases} L_1 = \sqrt{(x-x_1)^2 + (y-y_1)^2} \\ L_2 = \sqrt{(x-x_2)^2 + (y-y_2)^2} \end{cases} \tag{3.7}$$

Onde,

$$(x_1, y_1) = (0, h) \tag{3.8}$$

$$(x_2, y_2) = (b, h) \tag{3.9}$$

Logo,

$$\begin{cases} L_1 = \sqrt{x^2 + (y-h)^2} \\ L_2 = \sqrt{(x-b)^2 + (y-h)^2} \end{cases} \tag{3.10}$$

A equação (3.10) dá-nos a cinemática inversa do sistema da Figura 3.1.

3.1.2 – Sistema em três dimensões

Para perceber melhor o funcionamento do sistema é importante analisar a cinemática directa e indirecta. Na Figura 3.3, podemos ver o esquema da configuração em três dimensões que é objecto de estudo desta dissertação.

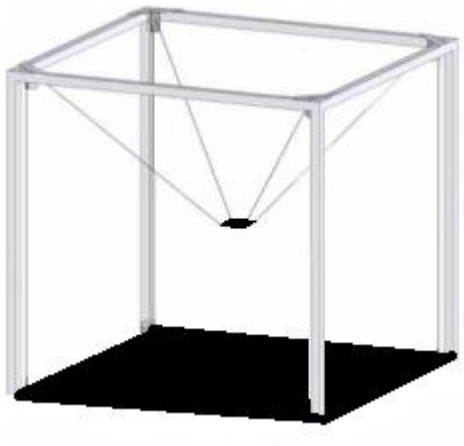


Figura 3.3: Sistema em três dimensões

Cinemática inversa

Como o espaço de trabalho é muito grande comparando com a distância entre os pontos onde os cabos são fixo na carga, podemos considerar que são fixos num único ponto. O comprimento dos cabos é dado pela distância entre a carga e o ponto onde o cabo é fixo. Logo, de um modo geral o comprimento do cabo é dado por:

$$L_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \tag{3.11}$$

Onde, Li é o comprimento do cabo i, (xi,yi,zi) são as coordenadas do ponto onde está fixo o cabo i, sendo que i=1,2,3,4 e (x,y,z) são as coordenadas da carga.

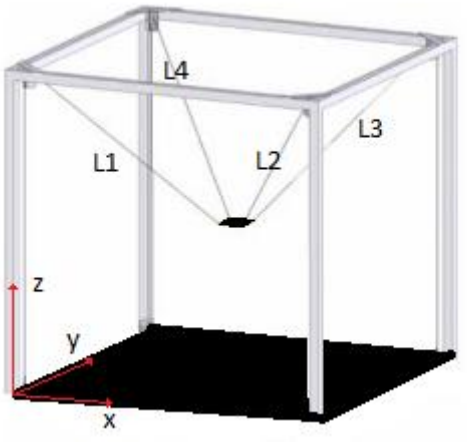


Figura 3.4: Eixo coordenado base do sistema 3D

Considerando o referencial da Figura 3.4, e que os pontos onde são fixos, os cabos formam um rectângulo entre eles a uma altura h, igual para todos. Temos,

$$(x_1, y_1, z_1) = (0,0,h) \tag{3.12}$$

$$(x_2, y_2, z_2) = (a,0,h) \tag{3.13}$$

$$(x_3, y_3, z_3) = (a,b,h) \tag{3.14}$$

$$(x_4, y_4, z_4) = (0,b,h) \tag{3.15}$$

Onde, *a* e *b* são o comprimento e largura do rectângulo formado pelos pontos onde são fixos os cabos. Substituindo estes valores na equação (3.11) obtemos a cinemática inversa para a configuração da Figura 3.4.

$$L_1 = \sqrt{x^2 + y^2 + (z-h)^2} \tag{3.16}$$

$$L_2 = \sqrt{(x-a)^2 + y^2 + (z-h)^2} \tag{3.17}$$

$$L_3 = \sqrt{(x-a)^2 + (y-b)^2 + (z-h)^2} \tag{3.18}$$

$$L_4 = \sqrt{x^2 + (y-b)^2 + (z-h)^2} \tag{3.19}$$

Cinemática directa

Resolvendo as equações anteriores em ordem a *(x,y,z)* obtemos a cinemática directa. É de notar que na configuração utilizada bastam três equações para definir a posição *(x,y,z)* da carga. Em termos práticos a posição da carga é definida por três dos cabos e o quarto cabo posiciona-se de forma a permitir a posição desejada. Considerando o seguinte sistema de equações,

$$\begin{cases} {L_1}^2 = x^2 + y^2 + (z-h)^2 \\ {L_2}^2 = (x-a)^2 + y^2 + (z-h)^2 \\ {L_3}^2 = (x-a)^2 + (y-b)^2 + (z-h)^2 \end{cases} \tag{3.20}$$

Com alguma manipulação algébrica facilmente obtemos o seguinte sistema,

$$\begin{cases} x^2 = {L_1}^2 - y^2 - (z-h)^2 \\ y^2 = {L_2}^2 - (x-a)^2 - (z-h)^2 \\ (z-h)^2 = {L_3}^2 - (x-a)^2 - (y-b)^2 \end{cases} \tag{3.21}$$

Substituindo a segunda equação do sistema (3.21) na primeira obtemos,

$$x^2 = {L_1}^2 - {L_2}^2 + (x-a)^2 + (z-h)^2 - (z-h)^2 \tag{3.22}$$

Resolvendo a equação anterior em ordem a *x* temos,

$$x = \frac{{L_1}^2 - {L_2}^2 + a^2}{2a} \tag{3.23}$$

Substituindo a terceira equação do sistema (3.21) na segunda obtemos,

$$y^2 = {L_2}^2 - (x-a)^2 - \left({L_3}^2 - (x-a)^2 - (y-b)^2\right) \tag{3.24}$$

Resolvendo a equação anterior em ordem a *y* temos,

$$y = \frac{{L_2}^2 - {L_3}^2 + b^2}{2b} \tag{3.25}$$

Resolvendo a primeira equação em ordem a *z* obtemos duas soluções possíveis,

$$z = h + \sqrt{{L_1}^2 - x^2 - y^2} \tag{3.26}$$

ou

$$z = h - \sqrt{{L_1}^2 - x^2 - y^2} \tag{3.27}$$

Uma vez que estamos a trabalhar com valores reais o valor dentro da raiz tem de ser sempre positivo. Tendo isso em conta e analisando a Figura 3.4 verificamos que a equação (3.26) não tem sentido neste caso, pois as soluções para a coordenada *z* seriam fora do espaço de trabalho.

Resumindo, para o sistema em estudo (Figura 3.4) a cinemática directa é definida por,

$$\begin{cases} x = \dfrac{{L_1}^2 - {L_2}^2 + a^2}{2a} \\ y = \dfrac{{L_2}^2 - {L_3}^2 + b^2}{2b} \\ z = h - \sqrt{{L_1}^2 - x^2 - y^2} \end{cases} \tag{3.28}$$

note-se que *z* ficou resolvido em termos de *x* e *y* por simplicidade.

3.2 – Dinâmica

O modelo dinâmico do sistema pode ser muito útil para acrescentar realismo à simulação e permitir um estudo mais rigoroso dos algoritmos a implementar. A dinâmica da plataforma pode ser descrita por um sistema de equações diferenciais [8]. Por simplificação a massa dos cabos é desprezada e os cabos são assumidos como rígidos, isto é, a elasticidade dos cabos é ignorada.

As equações da dinâmica do sistema podem ser calculadas usando o método de Lagrange [33]. A mecânica de Lagrange é uma formulação da mecânica clássica que combina a conservação do momento linear com a conservação da energia. Considerando o sistema com uma possível direcção de movimento, denotada pela variável de posição *x*. A energia cinética pode, dependendo do sistema de coordenadas usado, depender tanto da posição como da velocidade do movimento do objecto em questão, por isso podemos usar a seguinte notação *T(x,x')*. Da mesma forma, podemos escrever a energia potencial do sistema como *U(x,x')*. O lagrangiano é dado pela equação:

$$L(x, \dot{x}) = T(x, \dot{x}) - U(x, \dot{x}) \tag{3.29}$$

Onde *T* é a energia cinética total do sistema e *U* é a energia potencial total.

A dinâmica do sistema é determinada pela "equação de movimento", que é dada pela equação de Lagrange,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} = \frac{\partial L}{\partial x} \tag{3.30}$$

Sendo que, a energia cinética pode ser expressa por,

$$T = \frac{1}{2}mv^2 \tag{3.31}$$

Onde *m* é a massa do objecto em questão e *v* é a velocidade do mesmo. No caso em estudo, a expressão fica,

$$T = \frac{1}{2} m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) \tag{3.32}$$

A energia potencial pode ser expressa por,

$$U = m.g.h \tag{3.33}$$

Sendo que, *g* é a força gravítica e *h* é a altura do objecto. No caso em estudo será,

$$U = m.g.z \tag{3.34}$$

Substituindo a equação (3.29) na (3.30) obtemos,

$$\frac{d}{dt}\frac{\partial}{\partial \dot{x}}(T(x,\dot{x}) - U(x,\dot{x})) = \frac{\partial}{\partial x}(T(x,\dot{x}) - U(x,\dot{x})) \tag{3.35}$$

Analisando as equações (3.32) e (3.34), é fácil de verificar que,

$$\frac{\partial}{\partial x} T(x,\dot{x}) = 0 \tag{3.36}$$

e

$$\frac{\partial}{\partial \dot{x}} U(x,\dot{x}) = 0 \tag{3.37}$$

Combinando as equações anteriores e a equação (3.35), obtemos,

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{x}} = -\frac{\partial U}{\partial x} \tag{3.38}$$

Aplicando a equação anterior para o sistema em causa, obtemos o seguinte sistema de equações:

$$\begin{cases} \frac{d}{dt}(m\dot{x}) = 0 \\ \frac{d}{dt}(m\dot{y}) = 0 \\ \frac{d}{dt}(m\dot{z}) = -mg \end{cases} \tag{3.39}$$

Analisando o sistema de equações anteriores verificamos que só existe aceleração no eixo do *zz* e que tem sentido negativo. Se tivermos em conta que não estamos a considerar nenhuma força a actuar no sistema para além da força gravítica é fácil de verificar que este resultado faz todo o sentido. Podemos ainda confirmar este resultado aplicando a lei de Newton ao sistema.

Assumindo que,

$$F = \sum_{i=1}^{4} f_i \tag{3.40}$$

Onde *fi* representa a força exercida pelo cabo *i*, sendo que *i*=1,…,4 e é do tipo,

$$f_i = \begin{bmatrix} f_{i,x} \\ f_{i,y} \\ f_{i,z} \end{bmatrix} \tag{3.41}$$

Logo, F será do tipo,

$$F = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3.42}$$

Considerando a força resultante da tensão dos cabos na carga, a equação que descreve a dinâmica do sistema fica,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = \mathrm{F} \tag{3.43}$$

Aplicando a equação para o sistema em estudo, obtemos o seguinte sistema,

$$\begin{cases} f_x = \dfrac{d}{dt}(m\dot{x}) \\ f_y = \dfrac{d}{dt}(m\dot{y}) \\ f_z = \dfrac{d}{dt}(m\dot{z}) + mg \end{cases} \tag{3.44}$$

Com as equações (3.28) e (3.44), o sistema em estudo fica definido em termos de cinemática e dinâmica.

4 – Algoritmos para limitação do espaço de trabalho

4.1 – Simulação

Nos dias de hoje recorre-se muito à simulação para ajudar na implementação de novos sistemas. O facto de se criarem simuladores, pode trazer grandes vantagens em projectos de grande dimensão. O que inicialmente pode gastar algum tempo, para se poder ter um bom simulador que permita fazer teste fiáveis e conclusivos, pode ser compensado posteriormente pela facilidade com que se podem testar novas abordagens ao sistema a implementar. Outra grande vantagem da simulação é permitir ter várias pessoas a trabalhar no mesmo projecto com maior facilidade. Em alguns casos torna-se muito dispendioso fazer testes do sistema e por isso recorre-se a simulação sempre que possível.

No caso deste projecto a simulação foi importante por vários aspectos. Pode-se referir, a dificuldade em ter um local onde o sistema possa estar permanentemente instalado para se poder trabalhar. Visto que o sistema é para trabalhar em espaços muito amplos, como um estádio, torna-se difícil trabalhar constantemente no próprio sistema. Esta tarefa era ainda dificultada por existirem várias entidades a trabalhar no projecto, que apesar de estarem a trabalhar em partes diferentes precisam igualmente de acesso ao sistema. Um outro aspecto que tornou a simulação muito importante neste projecto foi a possibilidade de testar os algoritmos a implementar. Como se tratavam de algoritmos de alguma complexidade, a simulação foi uma mais-valia para comparar soluções e validar a solução final.

4.1.1 – Simulação em duas dimensões

Para simular o sistema em duas dimensões foi utilizado o *Matlab*. Como podemos ver na Figura 4.1, a carga é representada por um círculo vermelho e o vector velocidade desejado é representado por uma seta azul. Também a azul vemos uma semi-recta que representa um obstáculo. Foram construídas duas versões diferentes, numa o controlo da posição da carga é feito por meio de vectores de velocidade introduzidos por um joystick e na outra é seleccionada a posição para onde se quer deslocar a carga, utilizando o rato. Em ambas a versões o simulador permite introduzir os obstáculos com o auxílio do rato e é possível carregar os obstáculos usados anteriormente.

O algoritmo para evitar a colisão com os obstáculos foi baseado no declive da equação da recta. É feita a comparação entre o declive das equações das rectas formadas pelos cabos, com os declives das rectas formadas pelas arestas dos obstáculos, e com os declives das rectas formadas pelos pontos onde se fixam os cabos a cada um dos vértices dos obstáculos. Podemos dividir o algoritmo em duas verificações distintas, uma é quando a aresta a verificar tem um declive superior ao declive da recta que une o vértice mais alto dessa mesma aresta ao ponto onde se fixa o cabo (Figura 4.1), o outro caso é quando o declive da aresta é inferior (Figura 4.2). No primeiro caso, se

garantirmos que o vértice superior não gera colisão já é garantido que o vértice inferior não colide, ver Figura 4.1.

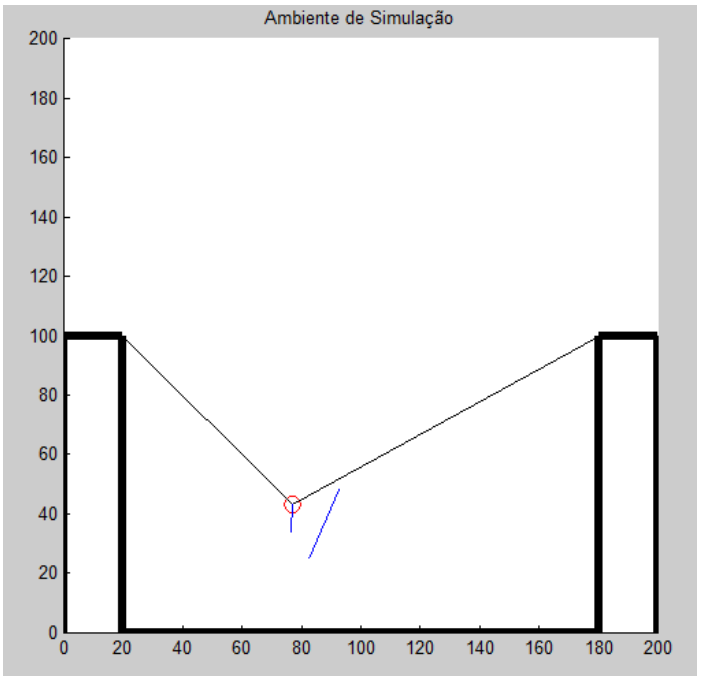


Figura 4.1: Simulador 2D que mostra um caso onde só é necessário verificar o vértice superior da aresta.

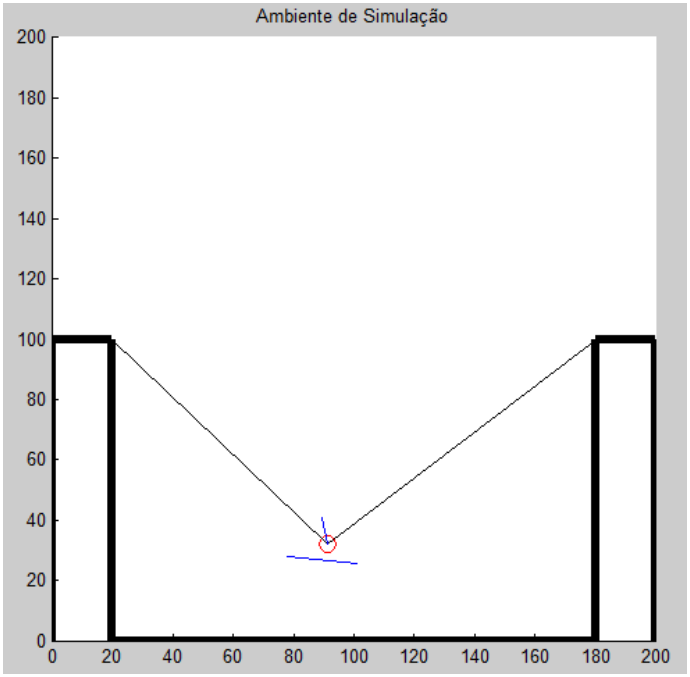


Figura 4.2: Simulador 2D que mostra um caso em que o declive da aresta é inferior ao declive formado pelos vértices e pelos pontos onde se fixam os cabos.

Sendo assim, para evitar colisão no primeiro caso, basta garantir que o declive do cabo é sempre inferior ao declive da recta que passa pelo vértice mais alto. Esta verificação só tem de ser feita para um dos cabos. Isto é, se a carga estiver à direita do vértice temos de garantir que o declive do cabo L_1 é inferior ao declive da recta que une o vértice em questão ao ponto onde se fixa o cabo L_1. Se a carga estiver à esquerda do vértice temos de garantir que o cabo L_2 tem declive inferior ao declive da recta formada pelo vértice e pelo ponto onde é fixo o cabo L_2.

Para o segundo caso temos três situações distintas, quando a carga se situa numa posição entre o valor *Xmáximo* e *Xmínimo* da aresta em questão e a posição em *y* é igual ou inferior ao valor de *y* do vértice mais alto da aresta (Figura 4.2), e quando a carga está à esquerda ou à direita da aresta em questão. Nas situações em que a carga está à esquerda ou à direita da aresta basta aplicar a solução descrita para o primeiro caso, descrito no parágrafo anterior, ao vértice mais próximo da carga. Para a situação em que a carga se encontra a uma altura igual ou inferior ao vértice mais alto da aresta e entre o *Xmáximo* e *Xmínimo* o declive da recta que passa pela carga e pelo vértice mais alto da aresta tem de ser inferior ao declive da aresta para evitar colisão.

Para tornar o algoritmo mais rápido e eficaz só são feitas verificações aos vértices e arestas que estejam a uma altura igual ou superior à posição actual da carga. É de notar que cada obstáculo é constituído, no mínimo, por dois pontos.

A nível de simulação temos de garantir ainda que a carga não passa os limites do espaço de trabalho. Isto é, a carga não pode passar os pontos onde são fixos os cabos. No caso real isto não é uma preocupação porque os cabos apenas permitem puxar uma carga e não empurrar.

No simulador em que é feito o controlo em posição é usado o método da força atractiva dos algoritmos de campos de potencial de robótica móvel [12]. A equação (4.1) traduz a distância entre o ponto de destino e o ponto actual em uma força de atracção.

$$F_A(i,j) = F_{ca}\left[\frac{x_t - x_0}{d_t}\hat{x} + \frac{y_t - y_0}{d_t}\hat{y}\right] \tag{4.1}$$

Onde, F_{ca} é a força constante atractiva, d_t é a distância entre a carga e o ponto de destino, x_0 e y_0 são as coordenadas da carga, x_t e y_t são as coordenadas do ponto de destino. A distância entre a carga e o ponto de destino é dada pela seguinte equação:

$$d_t = \sqrt{(x_t - x_0)^2 + (y_t - y_0)^2} \tag{4.2}$$

4.1.2 – Simulação em três dimensões

A simulação em três dimensões do sistema a implementar foi crucial na implementação dos algoritmos de detecção de obstáculos. Mais uma vez utilizou-se o *Matlab* para criar o simulador.

Para simplificar a introdução dos obstáculos e não sobrecarregar os sistemas optou-se por ter superfícies planas como obstáculos. Para trabalharmos polígonos tridimensionais, como por exemplo um paralelepípedo, o sistema ia ficar muito pesado e a introdução dos polígonos tornava-se algo complexa. Para além disso se alisarmos bem, uma superfície rectangular, quando definida com cuidado pode simular um obstáculo paralelepípedo, como podemos ver analisando a Figura 4.3.

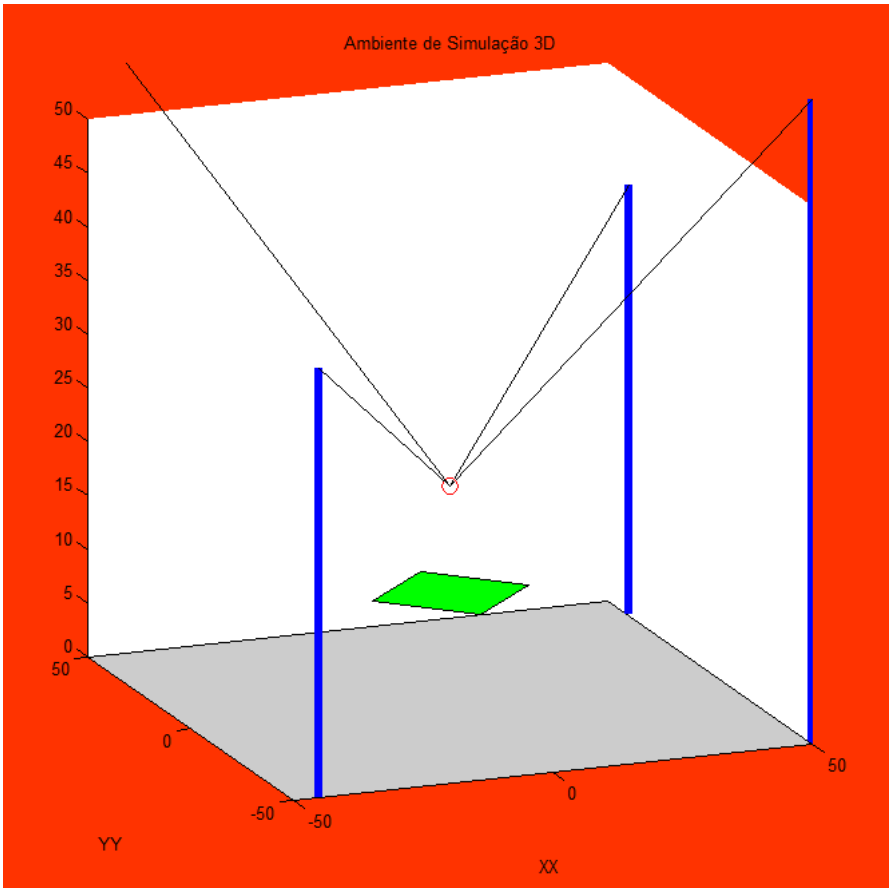


Figura 4.3: Superfície plana horizontal

Como podemos ver na Figura 4.3, apesar de o obstáculo ser constituído apenas por uma superfície plana, em termos práticos é equivalente a ter um paralelepípedo porque a existência dos cabos não permite deslocar a carga para posições que fiquem por baixo da superfície definida como obstáculo.

É importante notar que para um caso idêntico ao referido anteriormente em que o plano seja muito pequeno comparando com o espaço de trabalho, ou se quisermos

considerar o mesmo plano da Figura 4.3 mas com uma altura superior a situação referida já não se verifica. Isto é, nesse caso definir uma superfície plana não é suficiente para excluir a zona que esta por baixo da mesma. Devido ao reduzido tamanho do plano e à sua altura é possível passar a carga por baixo do mesmo sem tocar com os cabos na superfície. Para estes casos basta definir quatro superfícies planas laterais em conjunto com a superfície horizontal.

Na simulação em três dimensões os obstáculos podem ser definidos previamente num ficheiro *TXT* ou podem ser introduzidos durante a simulação com o auxílio do joystick. Por simplicidade estabeleceu-se dois tipos de superfícies planas para definir os obstáculos, com três ou quatro vértices. Por se tornar difícil definir quatro pontos durante a simulação que estejam todos no mesmo plano, os obstáculos introduzidos durante a simulação com o auxílio do joystick são definidos apenas com três pontos. Assim fica garantido que temos um plano, isto é, os três pontos são suficientes para definir um plano e não temos de ter a preocupação em marcar um quarto ponto no plano definido pelos três primeiros.

O algoritmo para evitar colisão com os obstáculos foi baseado na ideia de oclusão/intersecção, isto é, se imaginarmos os cabos como sendo feixes de luz queremos evitar que haja oclusão do mesmo entre a carga e o ponto onde o cabo em questão está fixo. Se imaginarmos os cabos como linhas, queremos evitar intersecção das linhas que representam os cabos com as superfícies que representam os obstáculos.

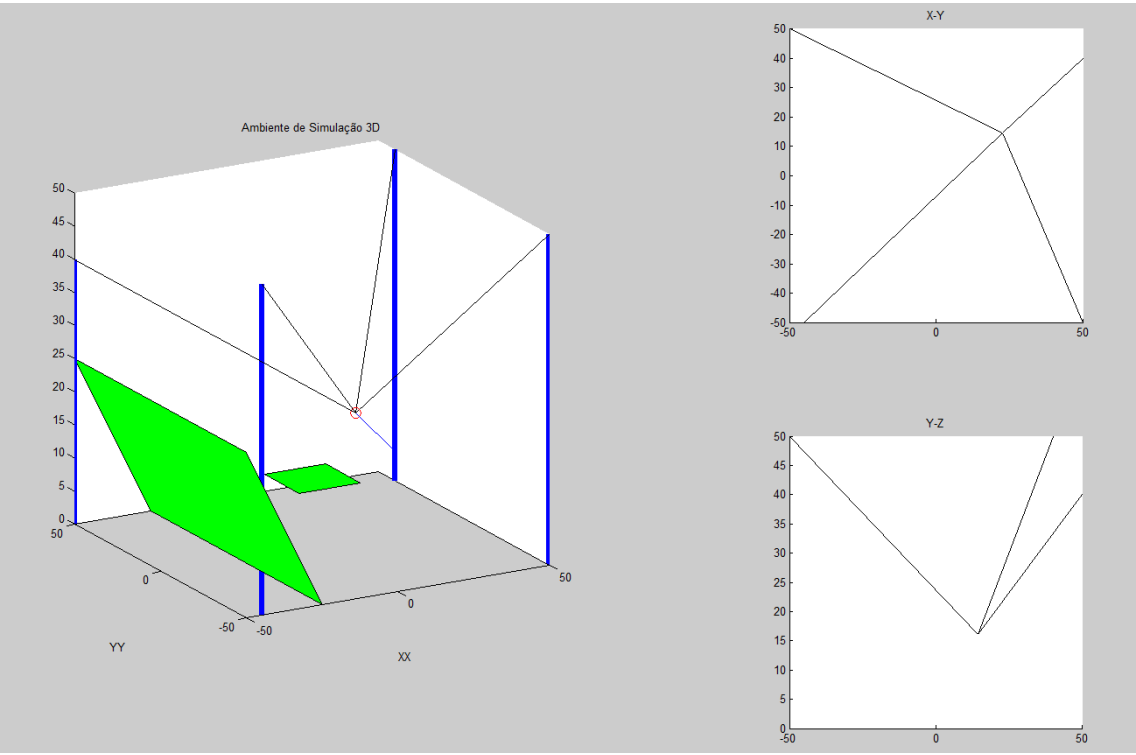


Figura 4.4: Ambiente de simulação em 3D

Para uma primeira abordagem foi utilizada a biblioteca *geom3d* disponível para usar em *Matlab* que permite manipular primitivas geométricas em três dimensões. A biblioteca *geom3d* para *Matlab* permite trabalhar e visualizar primitivas geométricas em três dimensões, como por exemplo pontos, linhas, planos, poliedros, etc. A biblioteca providencia funções de baixo nível para manipular primitivas geométricas em 3D, tornado mais fácil desenvolver algoritmos baseados em geometria mais complexos.

Tendo em conta que o simulador em três dimensões está limitado a um ecrã com visualização em duas dimensões optou-se por criar três janelas de visualização (Figura 4.4). Uma primeira janela onde é utilizado a função *plot3()* do *Matlab* para visualizar o ambiente de trabalho da carga tridimensional e duas janelas mais pequenas que permitem uma visualização em duas dimensões de duas das laterais do espaço de trabalho. As duas janelas mais pequenas permitem ter uma noção da posição da carga quando a visualização tridimensional não é muito clara.

4.2 – Algoritmos

Nesta secção serão apresentados os algoritmos implementados ao longo do projecto. Podemos referir como principais algoritmos implementados, detecção de zonas de colisão, introdução dos vértices das superfícies planas por qualquer ordem, e contornar obstáculos com trajectórias paralelas aos mesmos. É de notar que estes algoritmos têm de correr num *PLC* que é programado numa linguagem de texto estruturado (ST – Structured Text). Este facto requer a criação de algumas funções básicas para serem usadas nos algoritmos a implementar. Resumidamente, as operações básicas a implementar são funções para operações com matrizes e multiplicação vectorial.

4.2.1 – Detectar zonas de colisão

Para prevenir a colisão da carga ou dos cabos com os obstáculos é utilizado um método de prevenção, isto é, com base na velocidade da carga e com o comando introduzido pelo utilizador é feita a previsão da posição futura da carga e verifica-se se existe colisão nessa posição.

Como foi referido anteriormente para se testar a existência de colisão, verifica-se se existe intersecção entre algum dos segmentos de recta que definem os cabos e as superfícies planas que definem os obstáculos. Para poder verificar se existe intersecção entre um segmento e uma superfície plana, começou-se por verificar se existe intersecção entre o segmento de recta e o plano da superfície.

Para representar uma linha definida por dois pontos podemos usar a seguinte equação paramétrica,

$$P(s) = P_0 + s(P_1 - P_0) \tag{4.3}$$

Onde *s* é um número real, P_0 e P_1 são dois pontos da linha. É de notar que para $0<s<1$, $P(s)$ é um ponto no segmento de recta definido pelos pontos P_0 e P_1 onde *s* é a fracção da distância ao longo do segmento de recta. Analisando a equação (4.3) é fácil de verificar que para $s=0$ temos $P(s)=P_0$ e para $s=1$ temos $P(s)=P_1$. Isto é, o valor de *s* vai definir a distância ao ponto P_0. Se o valor de *s* for negativo, $P(s)$ vai ser um ponto fora do segmento do lado de P_0, se tivermos $s>1$ o $P(s)$ vai ser um ponto fora do segmento de recta do lado de P_1.

Considerando,

$$\boldsymbol{u} = P_1 - P_0 \tag{4.4}$$

Podemos escrever a equação (4.3) da seguinte forma,

$$P(s) = P_0 + s\boldsymbol{u} \tag{4.5}$$

Um plano pode ser representado de várias formas, uma forma simples de representar um plano tendo um ponto do plano e o seu vector normal é a equação (4.6).

$$\boldsymbol{n} \cdot (P - V_0) = 0 \tag{4.6}$$

Onde $\boldsymbol{n}$ é a normal ao plano e V_0 é um ponto pertencente ao plano. É de notar que qualquer ponto *P* pertencente ao plano satisfaz a equação.

Em três dimensões, para uma linha e um plano temos duas situações possíveis, ou a linha é paralela ao plano ou intersecta o plano num único ponto. Tendo isto em conta, para verificar se existe intersecção entre uma linha e um plano começamos por verificar se a linha é paralela ao plano. Para testar se uma linha é paralela ao plano basta testar se,

$$\boldsymbol{n} \cdot (P_1 - P_0) = 0 \tag{4.7}$$

Sendo que, $\boldsymbol{n}$ é o vector normal ao plano, P_0 e P_1 são os pontos que definem a linha. Se a condição se verificar a linha é paralela ao plano.

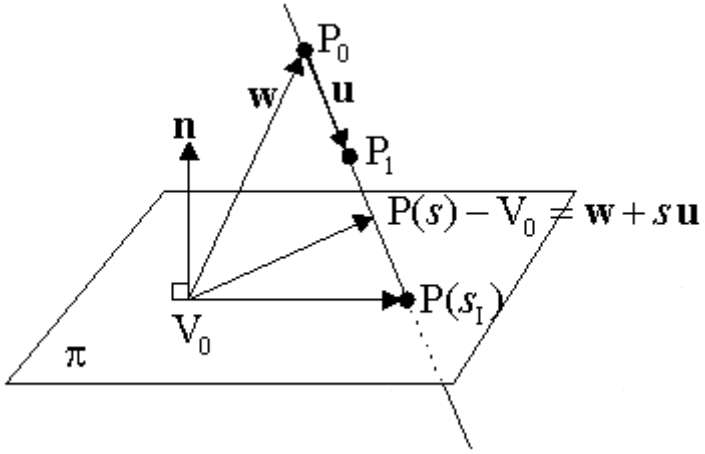


Figura 4.5: Intersecção entre uma linha e um plano.

Se tivermos perante um caso em que a linha não é paralela ao plano, significa que a linha intersecta o plano num ponto específico. Observando a Figura 4.5, no ponto de intersecção, o vector

$$P(s) - V_0 = \boldsymbol{w} + s\boldsymbol{u} \tag{4.8}$$

Onde

$$\boldsymbol{w} = P_0 - V_0 \tag{4.9}$$

É perpendicular ao vector $\boldsymbol{n}$. Isto é equivalente à condição dada pelo produto escalar,

$$\boldsymbol{n} \cdot (\boldsymbol{w} + s\boldsymbol{u}) = 0 \tag{4.10}$$

Resolvendo a equação em ordem a *s*, obtemos a seguinte expressão,

$$s_I = \frac{-\boldsymbol{n} \cdot \boldsymbol{w}}{\boldsymbol{n} \cdot \boldsymbol{u}} \tag{4.11}$$

Substituindo o S_I na equação que representa uma linha (4.3) obtemos o ponto de intersecção. É de notar que o valor de S_I dá-nos um ponto de intersecção entre a linha e o plano. Para verificarmos a intersecção do segmento de recta basta verificar se $0 \leq S_I \leq 1$.

Tendo o ponto de intersecção entre o segmento de recta e o plano que contêm a superfície, resta verificar se o ponto de intersecção está contido na superfície. Para fazer esta verificação optou-se por usar um algoritmo utilizado em computação gráfica para verificar se um ponto está contido num polígono em duas dimensões. Isso é possível

porque a superfície e o ponto de intersecção estão no mesmo plano, basta projectar a superfície e o ponto num dos três planos definidos pelo eixo coordenado.

O algoritmo para verificar se um ponto está dentro de um polígono em duas dimensões consiste em percorrer um raio semi-infinito na horizontal (aumentar x e fixar o y) para fora do ponto de teste, e contar o número de arestas que se atravessa. Se o número de intersecções for impar significa que o ponto está dentro do polígono, se for zero ou um número par é porque está fora.

A par do algoritmo que verifica colisão com obstáculos (anexo 1) foi desenvolvido um algoritmo que controla a distância aos limites do espaço de trabalho. Este controlo está dividido em três abordagens diferentes, as distâncias laterais, distância superior e distancia inferior (chão).

Para as laterais, tendo como base a equação do plano, podemos definir quatro planos para definir o espaço de trabalho em *XY*. Calculando os coeficientes da equação (4.12)

$$ax + by + cz + d = 0 \tag{4.12}$$

Sabendo que a distancia a um ponto é dada por:

$$D = \frac{|ax_1 + by_1 + cz_1 + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{4.13}$$

Onde (x_1,y_1,z_1) são as coordenadas do ponto. Se tivermos,

$$\sqrt{a^2 + b^2 + c^2} = 1 \tag{4.14}$$

Significa que a, b e c estão normalizados e a equação fica:

$$D = |ax_1 + by_1 + cz_1 + d| \tag{4.15}$$

Com os parâmetros da equação calculados podemos controlar a distância lateral no espaço de trabalho verificando o valor de *D* para a posição futura, como é feito com as superfícies planas que definem os obstáculos.

Como os pontos onde são fixos os cabos podem estar a alturas diferentes, é pouco provável que formem um plano. Por isso, para o topo foi considerado uma superfície dada por quatro pontos para se poder aproveitar o maior espaço de trabalho possível. Sendo assim, podemos definir uma superfície dada por quatro pontos com a seguinte equação:

$$z = ax + by + cxy + d \tag{4.16}$$

Considerando,

$$Ax = b \tag{4.17}$$

Sendo,

$$A = \begin{bmatrix} x1 & y1 & x1y1 & 1 \\ x2 & y2 & x2y2 & 1 \\ x3 & y3 & x3y3 & 1 \\ x4 & y4 & x4y4 & 1 \end{bmatrix}, \tag{4.18}$$

$$x = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{4.19}$$

e

$$b = \begin{bmatrix} z1 \\ z2 \\ z3 \\ z4 \end{bmatrix} \tag{5.20}$$

Podemos calcular os coeficientes *a*, *b*, *c* e *d*. De forma idêntica aos planos laterais é fácil de controlar a distância da carga aplicando as coordenadas na equação (4.16) e controlando a coordenada *z* (altura da carga). A função que inicializa os coeficientes das equações correspondentes ao planos laterais e a superfície do topo, pode ser consultada no anexo 2.

4.2.2 – Definição dos vértices dos obstáculos

No cálculo das intersecções entre linhas e superfícies planas é importante ter os planos bem definidos, isto é, a ordem de introdução dos vértices vai definir a orientação da normal. Para facilitar a tarefa do utilizador na introdução dos obstáculos criou-se um algoritmo que permite introduzir os vértices por qualquer ordem sem existir a preocupação com a orientação da normal.

Para definir um plano queremos que os vértices sejam introduzidos respeitando a regra da mão, isto é, os pontos serem introduzidos de forma rotacional em que a orientação da normal é dada pela regra da mão direita. A Figura 4.6 mostra um plano com a indicação da ordem de introdução dos pontos e o vector normal resultante.

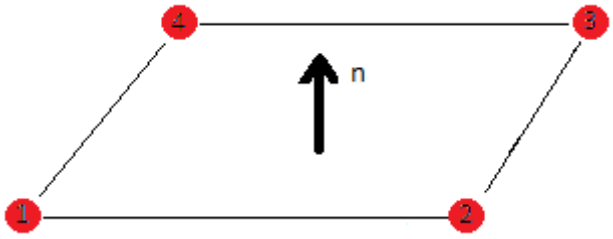


Figura 4.6: Plano com a normal marcada.

Inicialmente o algoritmo tem de verificar se os pontos estão ordenados de forma a percorrer os vértices da superfície de forma circular, isto é, o primeiro e terceiro ponto tem de estar em vértices diametralmente opostos, bem como o segundo ponto e o quarto. Para poder ordenar os pontos o algoritmo vai agrupar o primeiro e o terceiro ponto para criar um segmento de recta, e com os outros dois cria outro segmento de recta. Posto isto, temos três situações possíveis dependendo da ordem que o utilizador introduziu os pontos (Figura 4.7).

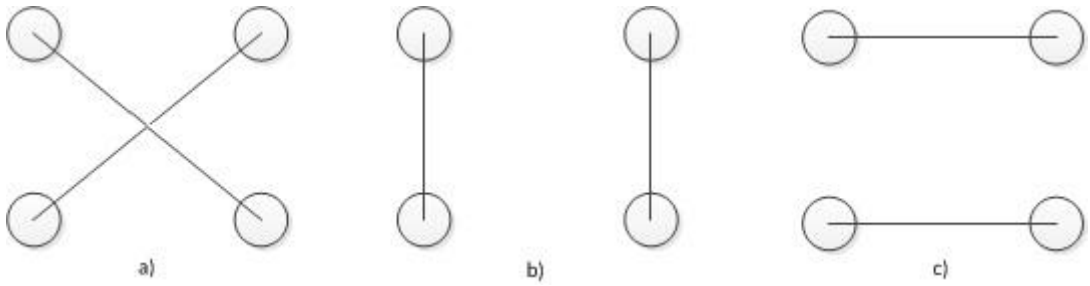


Figura 4.7: Configurações possíveis para introdução dos vértices.

Se tivermos perante a situação indicada da Figura 4.7(a) é porque os pontos já estão ordenados, pois o primeiro ponto e o terceiro já são diametralmente opostos. E ao termos o primeiro diametralmente oposto ao terceiro, os outros dois são obrigatoriamente opostos. Caso contrário é necessário testar outra combinação de pontos e verificar se já estamos no caso da Figura 4.7(a). Para verificar se estamos na situação indicada na Figura 4.7(a) basta testarmos se os dois segmentos definidos se intersectam.

De seguida temos de verificar a direcção da normal. Tendo em conta a estrutura do sistema (Figura 3.3) é fácil de verificar que a componente em *z* do vector normal à superfície deve ser sempre maior ou igual a zero. Isto deve-se ao facto de querermos a normal direccionada para a zona onde a carga pode trabalhar. No caso em que temos a componente em *z* da normal igual a zero significa que estamos perante um plano vertical, e num caso geral não é importante para onde está direccionada a normal nessa situação. É de notar que na implementação considerou-se que não seria necessário introduzir superfícies verticais devido a configuração do sistema. Esta consideração tem

por base a explicação feita anteriormente na explicação da simulação em três dimensões.

Em pseudocódigo o algoritmo é:

```
-Pegar no segmento (1) criado por ZonasProibidas[i].Ponto[1] e
ZonasProibidas[i].Ponto[3] e verificar se intersecta com o segmento
(2) criado por ZonasProibidas[i].Ponto[2] e
ZonasProibidas[i].Ponto[4];
Se não intersecta
            -Pegar no segmento (1) criado por
ZonasProibidas[i].Ponto[1] e ZonasProibidas[i].Ponto[2] e verificar se
intersecta com o segmento (2) criado por ZonasProibidas[i].Ponto[3] e
ZonasProibidas[i].Ponto[4];
      Se não intersecta
            -Pegar no segmento (1) criado por
      ZonasProibidas[i].Ponto[1] e ZonasProibidas[i].Ponto[4] e
      verificar se intersecta com o segmento (2) criado por
      ZonasProibidas[i].Ponto[3] e ZonasProibidas[i].Ponto[2];
      Fim_Se
Fim_Se
-Escolher um ponto do segmento 1 como primeiro vértice do plano;
-Escolher um dos pontos do segmento 2 como segundo vértice do plano;
-Escolher o outro ponto do segmento 1 como 3º vértice;
-Escolher o outro ponto do segmento 2 como 4º vértice;
Se a componente em Z da normal for negativa
            -Inverter a ordem dos vértices;
Fim_Se
```

Onde zonas proibidas é um array do tipo:

```
      ZonasProibidas: ARRAY[1..20] OF ST_PONTOS;
```

sendo *ST_PONTOS* uma estrutura do tipo:

```
TYPE ST_Pontos :
      STRUCT
                        Ponto: ARRAY[1..4] OF ST_XYZ;
      END_STRUCT
END_TYPE
```

e *ST_XYZ* é uma estrutura do tipo:

```
TYPE ST_XYZ :
      STRUCT
                         X: LREAL;
                        Y: LREAL;
                        Z: LREAL;
      END_STRUCT
END_TYPE
```

4.2.3 – Contornar obstáculos

Para permitir uma navegação mais simples perto de obstáculo foi criado um algoritmo que permite contornar os obstáculos. Com o modo de contornar obstáculos

activo, se o utilizador introduzir um vector velocidade no joystick que leva à colisão com um obstáculo, o algoritmo corrige esse vector para um vector que seja paralelo ao obstáculo.

Tendo em conta que estamos perante um espaço tridimensional, que para além da carga temos de ter em conta os cabos que a suportam e que os cabos têm uma configuração diferente para cada posição no espaço de trabalho, não é trivial criar um algoritmo que contorne os obstáculos. Por isso, fizeram-se várias abordagens até encontrar uma solução possível de implementar no sistema em questão.

1º Abordagem:

Calcular o plano feito pela aresta onde existe colisão com o ponto onde esse cabo é fixo. E calcular um vector velocidade paralelo ao plano calculado.

Notas práticas:

O método para contornar os obstáculos com base no plano criado pela aresta onde surge colisão e o ponto de fixação do cabo que colide não pode ser aplicado. Como a verificação de colisão é feita com base numa previsão da posição futura, a colisão que é detectada com mais frequência não é com a aresta do polígono mas sim com um ponto no meio do polígono. Podíamos calcular a aresta mais próxima a esse ponto, mas isso só resolvia o problema se conseguíssemos distinguir entre os casos:

(i) Uma intersecção que é originada por aproximação do cabo (Figura 4.8);
(ii) Uma intersecção por descer a câmara (Figura 4.9).

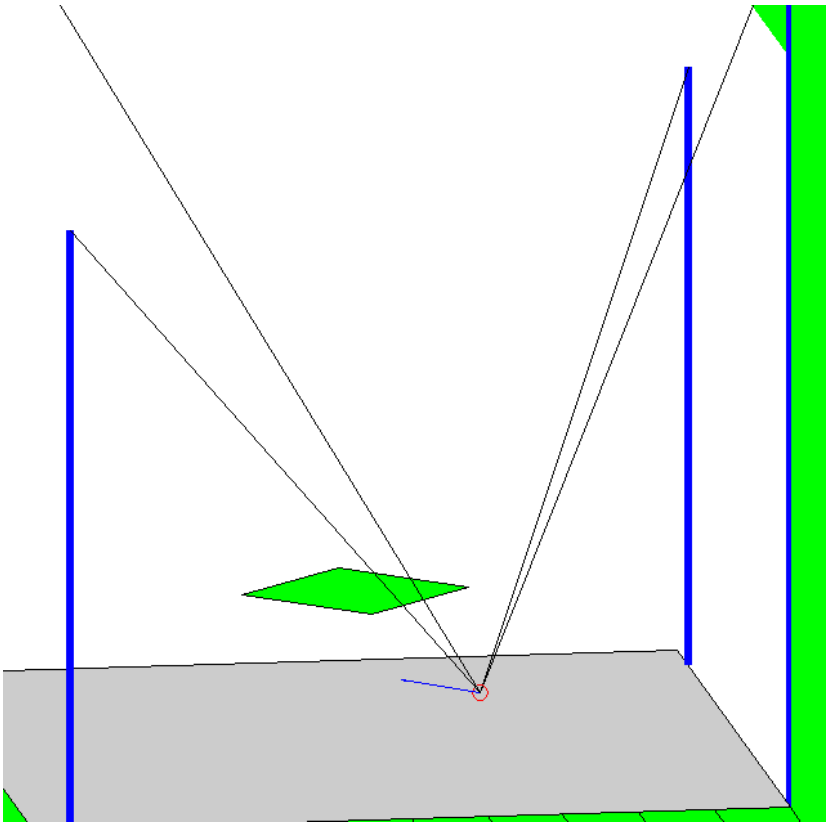


Figura 4.8: Intersecção originada por aproximação do cabo.

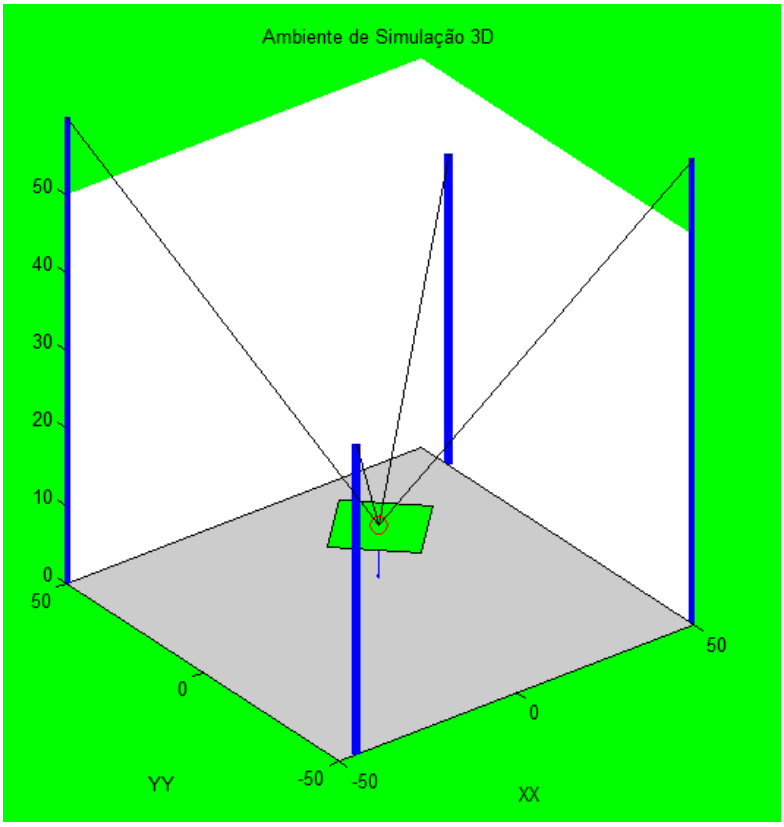


Figura 4.9: Intersecção por descer a câmara.

2º Abordagem:

Calcular o espaço de trabalho tendo em conta as zonas de exclusão (já a contar com os cabos) e tratar o problema com um algoritmo de robótica móvel (ex. Diagramas de Voronoi).

Notas Práticas:

Este método iria necessitar de uma algoritmo complexo para poder calcular o espaço de trabalho devido à diversidade de configurações possíveis. O cálculo do espaço de trabalho pode tornar-se muito pesado em termos de esforço computacional. Os algoritmos de robótica móvel, como é o caso dos Diagramas de Voronoi são para navegação em espaços livres e não são apropriados para contornar obstáculos da forma pretendida.

3º Abordagem:

Considerar dois tipos de colisão, um em que o vector velocidade é paralelo ao plano em que colide, e outro onde o vector velocidade não é paralelo.

Notas Práticas:

Esta abordagem permite ultrapassar o problema encontrado na primeira e tem um peso computacional aceitável. O algoritmo a implementar em pseudocódigo seria:

```
IF (detecta colisão) THEN
      IF (vector velocidade é paralelo ao plano) THEN
            (*Encontrar a aresta mais próxima e calcular o vector
            paralelo à aresta mais próximo do vector actual*)
      ELSE
            (*Calcular um vector paralelo ao plano*)
            IF (se o novo vector resulta em colisão) THEN
                  (*Encontrar a aresta mais próxima e calcular o vector
                  paralelo à aresta mais próximo do vector inicial*)
            END_IF
      END_IF
END_IF
```

4.2.4 – Funções implementadas

Devido à complexidade dos algoritmos implementados foi importante criar algumas funções para facilitar a implementação dos algoritmos no PLC. Umas das funções implementadas foi a inversão de matrizes. Como para os algoritmos implementados tínhamos no máximo matrizes de dimensão quatro optou-se por implementar uma solução que divide a matriz de dimensão quatro, em quatro matrizes de dimensão dois para fazer a inversão.

Considerando a matriz *M* dada por,

$$M = \begin{bmatrix} a_1 & \cdots & a_4 \\ \vdots & \ddots & \vdots \\ d_1 & \cdots & d_2 \end{bmatrix} \tag{4.21}$$

Podemos dividir a matriz M em quatro matrizes, isto é

$$M = \begin{bmatrix} P & Q \\ R & S \end{bmatrix} \tag{4.22}$$

Onde,

$$P = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}, \tag{4.23}$$

$$Q = \begin{bmatrix} a_3 & a_4 \\ b_3 & b_4 \end{bmatrix}, \tag{4.24}$$

$$R = \begin{bmatrix} c_1 & c_2 \\ d_1 & d_2 \end{bmatrix}, \tag{4.25}$$

e

$$S = \begin{bmatrix} c_3 & c_4 \\ d_3 & d_4 \end{bmatrix} \tag{4.26}$$

A matriz inversa de *M* pode ser calculada tendo em conta que,

$$M^{-1} = \begin{bmatrix} \bar{P} & \bar{Q} \\ \bar{R} & \bar{S} \end{bmatrix} \tag{4.27}$$

Onde,

$$\bar{P} = P^{-1} - (P^{-1}.Q).\bar{R} \tag{4.28}$$

$$\bar{Q} = -(P^{-1}.Q).\bar{S} \tag{4.29}$$

$$\bar{R} = -\bar{S}.(R.P^{-1}) \tag{4.30}$$

$$\bar{S} = (S - R.P^{-1}.Q)^{-1} \tag{4.31}$$

Com esta formulação, o determinante de M pode ser calculado usando a seguinte fórmula:

$$|M| = |P|.|S - R.P^{-1}.Q| \tag{4.32}$$

É de notar que para uma matriz ser invertível o seu determinante tem de ser diferente de zero. Para a formulação utilizada é fácil de verificar que se a matriz P não for invertível a matriz M também não pode ser invertível.

É importante referir que este método de particionar matrizes não pode ser aplicado em alguns casos como é referido em [24]. Mas para o efeito em que é utilizada não nos temos de preocupar com essas particularidades.

4.2.5 – Superfície plana com quatro vértices

É importante notar que para se criar uma superfície plana com quatro pontos a escolha do quarto ponto é limitada, isto é, para se definir o quarto ponto tem de se ter em conta o plano definido pelos três primeiros vértices. Uma forma simples de automatizar esta operação é escolher as coordenadas X e Y do quarto ponto e calcular a coordenada Z de forma que o ponto fique no plano definido pelos três primeiros vértices.

5 – Supervisão e comando

5.1 – OPC

Para ultrapassar a necessidade constante de instalar drivers, algumas empresas reuniram-se para desenvolver um modelo de comunicação padrão baseado na tecnologia OLE/DCOM para acesso a dados em tempo real dentro do sistema operativo Windows.

A comunicação OPC (OLE for Process Control) é um conjunto de protocolos padrões definidos pela fundação OPC [22] para a troca de informações entre aplicações de automação.

A tecnologia OLE (Object Linking and Embedding) foi desenvolvida pela Microsoft com o intuito de integrar diferentes aplicações dentro do sistema operativo Windows, de forma a solucionar problemas de desempenho do padrão DDE (Dynamic Data Exchange) utilizado anteriormente.

Basicamente, o DCOM (modelo objecto/componente distribuído) é um conjunto de definições para permitir a implementação de aplicações distribuídas em uma arquitectura cliente-servidor. Assim sendo, um cliente pode aceder a vários servidores em simultâneo e um servidor pode disponibilizar as suas funcionalidades para diferentes clientes ao mesmo tempo.

A criação do protocolo OPC trouxe diversas vantagens no desenvolvimento de produtos em automação industrial. Podemos referir como principais vantagens:

- Criação de um interface padrão de comunicação entre os servidores e clientes de dados em tempo real;
- Evitar a necessidade de drivers específicos para cada dispositivo;
- Melhoria em termos de desempenho das comunicações entre dispositivos de automação;
- Possibilidade de usar dispositivos de diversos fabricantes com grande facilidade;
- Facilidade em integrar aplicações para Windows em sistemas industriais;
- Evitar perder tempo a desenvolver interfaces e drivers de comunicação;
- Maior facilidade em desenvolver sistemas com comunicação em tempo real.

Hoje em dia, a utilização de comunicação OPC já pode ser encontrada em diversas aplicações industriais. Uma das aplicações comuns da comunicação OPC é para fins de visualização. Com a comunicação OPC torna-se fácil criar aplicações a correr em Windows que permitam manter sistemas industriais monitorizados.

Os dados OPC no cliente requerem algumas configurações, isto é, considerando o caso mais comum, que são os servidores de dados OPC (OPC Data Access), o cliente pode definir as seguintes configurações:

- Criação de grupos e itens OPC;
- Leitura Síncrona ou Assíncrona;
- Leitura de dados directo do dispositivo;
- Estado Activo/Inactivo;
- Leitura cíclica ou por mudança de estado;
- Banda Morta.

Os dados OPC são chamados de itens. Cada item pode ser de um tipo de dados diferente. Os itens são organizados por grupos OPC onde são definidas as suas características de leitura. As características de leitura dos itens são: taxa de actualização, estado activo/inactivo, banda morta e tipo de leitura (síncrona/assíncrona).

Num grupo com leitura síncrona, a leitura de dados depende de uma confirmação de execução antes de efectuar uma nova leitura. Para uma leitura assíncrona, não é necessário confirmação. A leitura dos dados pode ser feita directamente ao dispositivo ou pode ser feita da memória cache do servidor.

O estado de um grupo ou item no cliente pode ser alterado entre activo e inactivo, habilitando ou desabilitando a comunicação do mesmo.

Num cliente OPC pode-se definir se os dados do servidor são lidos de forma cíclica ou por mudança de estado. Na leitura cíclica, a aquisição dos dados é feita regularmente, isto é, os dados são actualizados quer sofram alteração quer não. Com a leitura por mudança de estado, o servidor envia para os clientes os itens que sofrerem alteração de estado ou os valores dos itens de um determinado grupo que ultrapasse o valor de banda morta.

A banda morta define os valores limites de transição para os itens de um determinado grupo.

É de notar que, o desempenho da comunicação OPC está próximo do desempenho apresentado por sistemas que utilizam drivers de comunicação específicos.

5.2 – Simulador 3D do MotionStudio

O software *MOVITOOLS MotionStudio* vem equipado com um sistema de simulação que permite correr os programas no *PLC* e utilizar o computador para visualizar o sistema implementado (Figura 5.1).

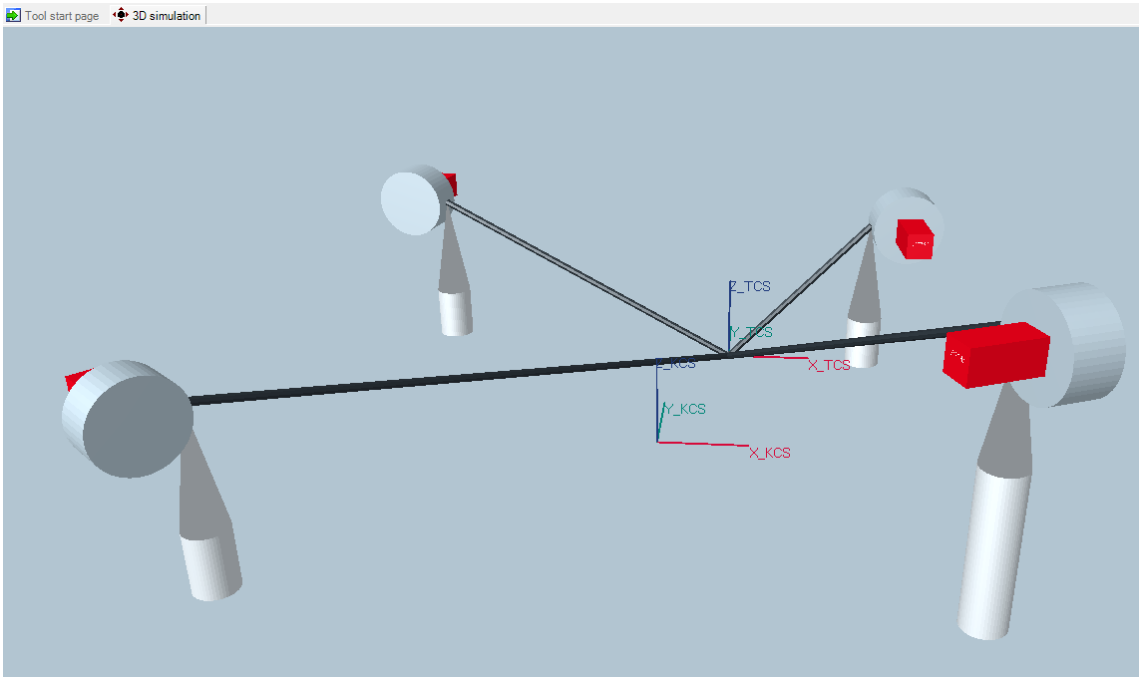


Figura 5.1 Simulador 3D do MotionStudio

5.3 – Visualização com base na comunicação OPC

O simulador do *MotionStudio* permite testar as implementações que vão sendo feitas, tornando-se numa grande vantagem no desenvolvimento do projecto. Mas, no que diz respeito à detecção de colisões, fica-se um pouco limitado. Isto é, o simulador do *MotionStudio* não permite visualizar os obstáculos, permite apenas visualizar os quatro eixos.

Como o *PLC* permite comunicação OPC, é possível desenvolver em *Matlab* uma aplicação que permita ligar ao *PLC* para criar um simulador idêntico ao já existente no software *MotionStudio*, permitindo visualizar os planos criados como obstáculos. Desta forma, torna-se possível testar os algoritmos de detecção de colisão com mais rigor e sem necessidade de proceder à instalação de toda a estrutura do sistema. Ou seja, basta ter um *Movi-PLC* com os joysticks ligados ao computador para poder testar a parte dos algoritmos relacionada com a detecção de obstáculos.

É de notar que, o *Matlab* possui uma toolbox de comunicação OPC que permite estabelecer ligação com os servidores OPC. A toolbox fornece as funções básicas para permitir trabalhar com comunicação OPC, tornando fácil a criação de aplicações com base neste tipo de comunicação. Antes de estabelecer a comunicação é necessário definir no *MotionStudio* as variáveis que vão estar disponíveis para serem acedidas por comunicação OPC. É também preciso configurar o servidor OPC para que possa ser acedido pelo *Matlab*. No anexo 3 podemos ver um pedaço de código que faz a ligação ao servidor OPC e lê uma variável do servidor.

6 – Resultados e conclusões

6.1 – Testes e resultados experimentais

Foi definido como métrica de validação e avaliação a distância entre a câmara e um obstáculo depois da paragem por detecção de colisão. Isto é, quando a posição futura da câmara leva à colisão, o sistema pára a câmara. A distância entre a câmara e o obstáculo, depois da câmara estar parada, foi definida como métrica de validação e avaliação.

Para poder validar os resultados obtidos foi necessário efectuar vários testes em variadas condições. Os testes efectuados foram:

- Deslocar a câmara a diferentes velocidades (por exemplo: velocidade máxima) na direcção do obstáculo;
- Deslocar a câmara paralelamente a um obstáculo e mudar o sentido do movimento em direcção ao obstáculo;
- Colocar a câmara perto do obstáculo e tentar arrancar na direcção do mesmo.

Os resultados finais demonstraram ser possível posicionar a câmara com uma distância ao obstáculo próxima de meio metro, variando mais ou menos 20 centímetros. Estes resultados foram obtidos tendo em conta uma velocidade máxima da câmara de 10 m/s e uma aceleração e desaceleração de 5 m/s^2. Nas tabelas seguintes é possível ver alguns dos resultados obtidos em testes efectuados.

Tabela 1: Resultados de deslocamento em X

Velocidade (mm/s)	Ponto Paragem (mm)	Distância (mm)	Variação (mm)
7200	48372	628	44,00
7200	48564	436	148,00
7200	48073	927	343,00
4000	48781	219	365,00
4000	48255	745	161,00
4000	48451	549	35,00
Média		584,00	182,67

Tabela 2: Resultados de deslocamento em Y

Velocidade (mm/s)	Ponto Paragem (mm)	Distância (mm)	Variação (mm)
7200	48572	428	203,00
7200	48472	528	103,00
7200	48079	921	290,00
4000	48263	737	106,00
4000	48481	519	112,00
4000	48347	653	22,00
Média		631,00	139,33

Tabela 3: Resultados de deslocamento em Z

Velocidade (mm/s)	Ponto Paragem (mm)	Distância (mm)	Variação (mm)
7200	43064	936	366,33
7200	43663	337	232,67
7200	43575	425	144,67
4000	43375	625	55,33
4000	43453	547	22,67
4000	43452	548	21,67
Média		569,67	140,56

De forma a obter os resultados demonstrados nas tabelas anteriores, posicionou-se a câmara a uma distância do obstáculo suficientemente grande para a poder estabilizar a uma velocidade constante antes de chegar à zona de segurança. No caso das tabelas 1 e 2 o obstáculo estava na posição 49000 mm e na tabela 3 o obstáculo estava na posição 44000 mm. Nas tabelas podemos ver a distância ao obstáculo para testes a diferentes velocidades.

Para o contorno de obstáculos a métrica de validação e avaliação para além da distância ao obstáculo é a variação entre o vector velocidade introduzido pelo utilizador e o vector resultante. Os testes feitos para este caso foram idênticos aos descritos anteriormente, sendo que as distâncias, da câmara ao obstáculo, variam de um a dois metros. A variação do vector velocidade torna-se mais difícil de avaliar, mas uma vez que o vector calculado é proporcional à componente paralela ao obstáculo do vector introduzido pelo utilizador, a variação do vector velocidade é mínima.

6.2 – Conclusões

Ao longo deste trabalho foram desenvolvidos algoritmos para evitar colisão com obstáculos em robôs paralelos conduzidos por cabos. Estes algoritmos tornaram o sistema mais robusto, permitindo uma navegação mais fácil na medida em que o utilizador não tem que estar preocupado se os cabos ou a carga vão colidir com algum obstáculo. Os algoritmos desenvolvidos têm por base a configuração geométrica das plataformas paralelas conduzidas por cabos.

Apesar do desenvolvimento feito ter sido para o caso particular do sistema OmniCam4sky, é possível aplicar os algoritmos desenvolvidos para outros sistemas com configurações idênticas. Isto é, os algoritmos desenvolvidos podem ser facilmente alterados para outras configurações de robôs paralelos conduzidos por cabos.

A comunicação OPC demonstrou ser uma ferramenta simples de utilizar e com elevado potencial. O facto de se ter explorado a comunicação OPC permitiu fazer uma abordagem ao sistema mais pormenorizada, o que tornou possível utilizar as potencialidades do *Matlab* para testar os algoritmos implementados no *Movi-PLC*.

6.3 – Trabalho futuro

Para um estudo mais rigoroso dos sistemas paralelos conduzidos por cabos, podia ser feito um estudo sobre o arco feito nos cabos que sustentam a carga. O facto de se ignorar este efeito vai trazer alguns erros na posição final da carga. Apesar destes erros serem insignificantes para o sistema em estudo, podem fazer diferença para sistemas que necessitem de uma maior precisão.

Uma abordagem que poderia ser interessante era estudar uma forma automática de introduzir os obstáculos do espaço de trabalho. Isto é, introduzir sensores que permitissem o reconhecimento automático das zonas de colisão.

Com o estudo feito a nível da comunicação OPC, seria interessante criar um interface para o utilizador, que permitisse uma forma fácil de introduzir os obstáculos, bem como outras funcionalidades para melhorar o interface Homem Máquina.

Referências

[1] R. Verhoeven, M. Hiller and S. Tadokoro (1998) "Workspace, Stifeness, Singularities and Classification of Tendon-Driven Stewart Platforms"

[2] R. Verhoeven and M. Hille (2000) "Estimating the controllable workspace of tendon-based Stewart Platforms"

[3] Samir Lahouar, Erika Ottaviano, Said Zeghoul, Lotfi Romdhane and Marco Ceccarelli (2009) "Collision free path-planning for cable-driven parallel robots"

[4] Damien Chablat, Erika Ottaviano and Guillaume Moroz (2011) "A Comparative Study of 4-Cable Planar Manipulators Based on Cylindrical Algebraic Decomposition"

[5] Ali Ghasemi, Mohammad Eghtesad and Mehrdad Farid (2010) "Neural Network Solution for Forward Kinematics Problem of Cable Robots"

[6] E. Ottaviano (2008) "Analysis and Design of a four-cable-driven parallel manipulator for a planar and spatial tasks"

[7] Derek McColl (2009) "Workspace generation for wire-actuated parallel manipulators"

[8] Vladirmir Gordievsky (2008) "Design and Control of Robotic Cable-Suspended camera system for operation in 3D Industrial Environment"

[9] Abdullah Basar Alp (2001) "Cable-Suspended Parallel Robots"

[10] Cícero dos Santos Mendes Lima Ribeiro (2010) "Análise da plataforma de Stewart accionada por cabos para grandes espaços de trabalho"

[11] Alexandre Back e Travi (2009) "Plataforma de Stewart accionada por cabos"

[12] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki and Sebastian Thrun (2005) "Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents series)"

[13] Dan Sunday, "Geometry Algorithms", SoftSurfer; http://softsurfer.com/algorithm_archive.htm, (acesso em 17 de Outubro 2011).

[14] W. Randolph Franklin, "Point Inclusion in Polygon Test"; http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html#The C Code (acesso em 17 de Outubro 2011).

[15] Sew-Eurodrive Portugal http://www.sew-eurodrive.pt/, (acesso em 20 de Janeiro 2012)

[16] Nicholas, Dagalakis, James S, Kenneth, James D, Tsung-Ming, Hoosh, Roger (1989) “Robor crane Technology”

[17] National Institute of Standards and Technology
http://www.nist.gov/el/isd/ , (acesso em 20 Janeiro 2012)

[18] Manual “Controlador MOVI-PLC advanced DHE21B/DHF41B/DHR41B” (Edição 04/2008)

[19] Manual Sew “Sistema I/O MOVI-PLC” (Edição 07/2007)

[20] Manual Sew “MOVITOOLS MotionStudio” (Edição 08/2009)

[21] Marcos de Oliveira Fonseca (2002), "Comunicação OPC - Uma abordagem prática"

[22] The Interaperability Standard for Industrial Automation & Other Related Domains, “OPC Foundation”
http://www.opcfoundation.org, (acesso em 20 Janeiro)

[23] Mathworks
http://www.mathworks.com/, (acesso em 20 Janeiro)

[24] Freevec.org, “Inverse of Matrix 4x4 using partitioning”
http://freevec.org/function/inverse_matrix_4x4_using_partitioning, (acesso em 20 Janeiro)

[25] Kawamura, S. and Ito, K. (1993) “A New Type of Master Robot for Teleopertaion Using a Radial Wire Drive System”.

[26] Kino, H., Miyazano, H., Won, C. and Kawamura, S., (1997) “Realization of Large Work Space using Parallel Wire Drive Robots”.

[27] Roberts, R. G., Graham, T., and Lippitt, T., (1998) “On the Inverse Kinematics, Statics, and Fault Tolerance of Cable-Suspended Robots”.

[28] Stump, E. and Kumar, V., (2006) “Workspaces of Cable-Actuated Parallel Manipulators”.

[29] Wikipedia, “Farkas' lemma”
http://en.wikipedia.org/wiki/Farkas'_lemma, (acesso em 20 Janeiro)

[30] Dagalakis, N., Albus, J. S., Wang, B. L., Unger, J., and Lee, J. D., (1988) “Stiffness of a Parallel Link Robot Crane for Shipbuilding Applications”.

[31] PLCopen for efficiency in automation, “Introduction into IEC 61131-3 Programming Languages”
http://www.plcopen.org/pages/tc1_standards/iec_61131_3/ , (acesso em 20 Janeiro)

[32] Fazendo Matemática, “Demonstração do Teorema de Heron” http://www.fazendomatematica.com/2010/09/demonstracao-do-teorema-de-heron.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+blogspot%2FlpgrS+%28Fazendo+Matem%C3%A1tica%29, (acesso em 20 Janeiro)

[33] Worldforge , “Lagrangian Mechanics Made Simple” http://www.worldforge.org/project/newsletters/May2002/LMMS_index, (acesso em 20 Janeiro)

[34] Korayem, M.H., Bamdad, M. and Saadat, M., (2007), “Workspace Analysis of Cable-Suspended Robots with Elastic Cable”.

[35] Wischnitzer, Y., Shvalb, N. and Shoham, M., 2008, “Wire-driven Parallel Robot: Permitting Collisions Between Wires”.

[36] Group of Robots & Intelligent Machines, “Robot Trepador de Estructura Paralela” http://sade.disam.etsii.upm.es/trepa/es/project.asp, (acesso em 20 Janeiro)

[37] Escola de Aviação Civil, "Instrutores de Simulador" http://www.stsrj.com.br/, (acesso em 20 Janeiro)

Apêndice

Anexo 1 – Função Intersecção

```
FUNCTION Intersect3D_SegPla : P_int
VAR_INPUT
      var_x: REAL;
      var_y: REAL;
      var_z: REAL;
END_VAR
VAR
      aux_u: ST_XYZ;
      aux_w: ST_XYZ;
      aux_d: ST_XYZ;
      Denominador: REAL;
      Numerador: REAL;
      sI: REAL;
      II: INT;
      vertx: ARRAY[1..4] OF REAL;
      verty: ARRAY[1..4] OF REAL;
      KK: INT;
      JJ: INT;
      dis_flag: REAL;
END_VAR

IF ProcessaFronteiras(var_x, var_y, var_z) <> 0 THEN
(*Se devolver um valor diferente de zero é porque existe colisão com
as "paredes" ou com o plano superior*)
      Intersect3D_SegPla.Flag :=1;
      RETURN;
END_IF

FOR KK:=1 TO 4 BY 1 DO
      aux_u.X := var_x-Pos_Cabo[KK].X;
      aux_u.Y := var_y-Pos_Cabo[KK].Y;
      aux_u.Z := var_z-Pos_Cabo[KK].Z;

      FOR II:=1 TO NumDePlanos BY 1 DO
            aux_d.X := var_x-ZonasProibidas[II].Ponto[1].X;
            aux_d.Y := var_y-ZonasProibidas[II].Ponto[1].Y;
            aux_d.Z := var_z-ZonasProibidas[II].Ponto[1].Z;
            dis_flag:=dot(Normal_Plan[II],aux_d);
            IF(dis_flag < 0)THEN
                  Intersect3D_SegPla.Flag :=1;
                  RETURN;
            END_IF

            Intersect3D_SegPla.Flag :=-1;(*por segurança inicia-se a
Flag a -1*)

            aux_w.X := Pos_Cabo[KK].X-ZonasProibidas[II].Ponto[1].X;
            aux_w.Y := Pos_Cabo[KK].Y-ZonasProibidas[II].Ponto[1].Y;
```

```
            aux_w.Z := Pos_Cabo[KK].Z-ZonasProibidas[II].Ponto[1].Z;

            Denominador := dot(Normal_Plan[II], aux_u);
            Numerador := -1*dot(Normal_Plan[II], aux_w);

            IF (ABS(Denominador)<0.00000001) THEN
                  IF (Numerador=0) THEN(*Segmento esta no plano*)
                        Intersect3D_SegPla.Flag :=2;
                  ELSE
                        Intersect3D_SegPla.Flag :=0;
                  END_IF
            ELSE
                  (*Se o segmento nao for paralelo ao plano calcula o
sI correspondente
                  da formula P(s)=P0+s(P1-P0) que nos dá o ponto de
intersecção*)
                  sI := Numerador/Denominador;
                  IF (sI<0 OR sI>1) THEN
                  (*não existe intersecção (a intersecção esta fora do
segmento,
                  isto é, intersecta a linha mas nao o segmento)*)
                        Intersect3D_SegPla.Flag :=0;
                  ELSE
                        Intersect3D_SegPla.Ponto.X :=
Pos_Cabo[KK].X+sI*aux_u.X;
                        Intersect3D_SegPla.Ponto.Y :=
Pos_Cabo[KK].Y+sI*aux_u.Y;
                        Intersect3D_SegPla.Ponto.Z :=
Pos_Cabo[KK].Z+sI*aux_u.Z;
                        Intersect3D_SegPla.Flag :=1;
                  END_IF
            END_IF

            IF Intersect3D_SegPla.Flag = 1 THEN
                  (*Cria tabela com vertices x e outra para y*)
                  FOR JJ:=1 TO 4 BY 1 DO
                        vertx[JJ] := ZonasProibidas[II].Ponto[JJ].X;
                        verty[JJ] := ZonasProibidas[II].Ponto[JJ].Y;
                  END_FOR
                  IF
InPoly(4,vertx,verty,Intersect3D_SegPla.Ponto.X,Intersect3D_SegPla.Pon
to.Y) THEN
                        Intersect3D_SegPla.Flag :=1;
                        RETURN;
                  ELSE
                        Intersect3D_SegPla.Flag :=-1;
                  END_IF
            END_IF
      END_FOR;
END_FOR
```

Anexo 2 – Função Inicializa Fronteiras

```
FUNCTION InitFronteiras : INT
(* Determina os parametros [a,b,c,d] dos planos de fronteira *)
VAR_INPUT
      Cabo: ARRAY[1..4] OF ST_XYZ;
END_VAR
VAR
      idx: INT;

      jdx: INT;


      nab: REAL;

      P: ARRAY[1..3] OF ST_XYZ;
      plano_aux: ST_PLANO;
      matA: ARRAY [1..4,1..4] OF REAL;
      aux_var: Mat_4x4;
END_VAR
(* Globais


      Lat: ARRAY[1..4] OF ST_PLANO;

      Top: ARRAY [1..4] OF REAL;
*)

(*
Determina equações dos planos verticais definidos pelos pontos de
apoio Cabo
      a(i).x+b(i).y+d(i)=0
*)
jdx := 4;

FOR idx := 1 TO 4 BY 1 DO
      Lat[jdx].a := Cabo[idx].y - Cabo[jdx].y;
      Lat[jdx].b := Cabo[jdx].x - Cabo[idx].x;
      Lat[jdx].c := 0;
      Lat[jdx].d := -Cabo[jdx].y*Lat[jdx].b - Cabo[jdx].x *
Lat[jdx].a;
      nab := SQRT(Lat[jdx].a*Lat[jdx].a + Lat[jdx].b*Lat[jdx].b);

Lat[jdx].a := Lat[jdx].a / nab;
      Lat[jdx].b := Lat[jdx].b / nab;
      Lat[jdx].d := Lat[jdx].d / nab;
      jdx := idx;
END_FOR

(*Superficie superior*)

(* Interpolacao
 z = ax + by + cxy + d *)
```

```
FOR idx:=1 TO 4 BY 1 DO
      matA[idx,1]:=Cabo[idx].X;
      matA[idx,2]:=Cabo[idx].Y;
      matA[idx,3]:=Cabo[idx].X*Cabo[idx].Y;
      matA[idx,4]:=1;
END_FOR

(*coeficientes da superficie*)
Top[1]:=0;
Top[2]:=0;
Top[3]:=0;
Top[4]:=0;
aux_var:=inv4x4(matA);
IF aux_var.d<>0 THEN (*Se o det(A)<>0 logo a matriz é invertivel*)
      FOR idx:=1 TO 4 BY 1 DO
            Top[1]:=Top[1]+matA[1,idx]*Cabo[idx].Z;
            Top[2]:=Top[2]+matA[2,idx]*Cabo[idx].Z;
            Top[3]:=Top[3]+matA[3,idx]*Cabo[idx].Z;
            Top[4]:=Top[4]+matA[4,idx]*Cabo[idx].Z;
      END_FOR
ELSE(*No caso da matriz ser não invertível vamos considerar para o
topo um plano horizontal com Z igual à altura do ponto de fixação mais
baixo.*)
      Top[4]:=MIN(MIN(MIN(Cabo[3].Z,Cabo[4].Z),Cabo[2].Z),Cabo[1].Z);
END_IF
```

Anexo 3 – Exemplo de leitura de uma variável para Matlab usando OPC

```
%Comunicação por OPC com Movi-PLC

%mostras os servidores disponiveis
hostInfo = opcserverinfo('localhost')

%mostra o ID dos servidores
allServers = hostInfo.ServerID'

%associa o servidor a uma variavel
da = opcda('localhost', 'CoDeSys.OPC.02')

%conecta ao servidor
connect(da)

%adiciona um grupo (de variaveis)
grp=addgroup(da)

%mostra as variaveis disponiveis
serveritems(da)

%associa a variavel ao grupo
itm=additem(grp, '.NumdePlanos')

%le o grupo (de variaveis)
read(grp)
```