UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

TRABALHO DE CONCLUSÃO DE CURSO

THIAGO ALBERICI ROBERTO

CÁLCULO DA CINEMÁTICA INVERSA DO ROBÔ MANIPULADOR PUMA 560 UTILIZANDO ALGORITMOS GENÉTICOS

São Carlos
2012

THIAGO ALBERICI ROBERTO

CÁLCULO DA CINEMÁTICA INVERSA DO ROBÔ MANIPULADOR PUMA 560 UTILIZANDO ALGORITMOS GENÉTICOS

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia De Computação com
ênfase em Robótica

ORIENTADOR: Valdir Grassi Jr.

São Carlos
2012

Roberto, Thiago Alberici
R639c Cálculo da cinemática inversa do robô manipulador puma 560 utilizando algoritmos genéticos / Thiago Alberici Roberto; orientador Valdir Grassi Jr.. São Carlos, 2012.

Monografia (Graduação em Engenharia de Computação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012.

1. Robôs Manipuladores. 2. Puma 560. 3. Cinemática Inversa. 4. Algoritmos Genéticos. I. Título.

FOLHA DE APROVAÇÃO

Nome: Thiago Alberici Roberto

Título: "Cálculo da Cinemática Inversa de Robô Manipulador PUMA 560 Utilizando Algoritmos Genéticos"

Trabalho de Conclusão de Curso defendido em 30 / 11 / 2012 .

Comissão Julgadora:	**Resultado:**
Prof. Dr. Fernando Santos Osório SSC/ICMC/USP	APROVADO
Prof. Associado Marcelo Becker SEM/EESC/USP	APROVADO

Orientador:
Prof. Dr. Valdir Grassi Júnior - SEL/EESC/USP

Coordenador pela EESC/USP do Curso de Engenharia de Computação:

Prof. Associado Evandro Luís Linhari Rodrigues

"Tenha em mente que tudo que você aprende na escola é trabalho de muitas gerações. Tudo isso é posto em sua mão como sua herança para que você receba-a, honre-a, acrescente a ela e, um dia, fielmente, deposite-a nas mãos de seus filhos".

Albert Einsten, The World As I See It

Agradecimentos

A todos os professores que eu tive na graduação, que me proporcionaram conhecimento para poder fazer esse e todos os projetos feitos durante a graduação. Aos funcionários e outros professores que de alguma forma deram suporte aos alunos trabalhando para que o curso esteja sempre em melhoria.

Ao professor Valdir Grassi Jr. Pelo apoio, incentivo e orientação durante o desenvolvimento desse projeto.

A minha família que sempre me apoiou e me deu forças em todos os projetos da minha vida. Principalmente meus pais (Edvaldo e Lizete) e meu irmão (Vinicius). Sem eles, nada disso teria sido possível. Agradeço também a Camila, que me acompanhou durante todos esses anos de faculdade, também me apoiando em todas as minhas decisões.

Aos amigos que durante os anos de faculdade, dividiram a república comigo, sempre proporcionando um ambiente alegre e de boa convivência. E principalmente, agradeço a todos os meus colegas de sala. Sem eles, o curso nunca teria sido o mesmo. Noites de estudo, de festa, de confraternização. Tudo isso foi muito importante não só na minha formação acadêmica, mas principalmente na minha formação como pessoa. Obrigado!

Resumo

Resolver a cinemática de um robô manipulador é um cálculo de grande importância na área de robótica, uma vez que esses robôs são usados em escala comercial e em várias pesquisas científicas. No entanto, esse é um cálculo que pode ser de grande complexidade. Este trabalho tem como objetivo propor uma maneira alternativa para o cálculo da cinemática inversa de robôs manipuladores. Como referência experimental para esse estudo, tomou-se o robô Puma 560 e foi implementado um sistema evolutivo baseado em algoritmos genéticos na linguagem C. Para os testes do método proposto, os resultados foram divididos em duas principais categorias (cálculo da posição final do manipulador e cálculo da posição e orientação final do manipulador). O método proposto foi analisado quanto à precisão e quanto ao tempo de processamento. Através de médias e desvio-padrão de uma bateria de 100 testes para cada configuração, pode-se concluir que o método é muito eficaz para exemplos onde só a posição final é considerada. Já para quando a complexidade do problema aumenta muito (considerando a orientação do manipulador) o algoritmo mostrou-se apenas razoável. Para tal cálculo, chegou-se a conclusão que em trabalhos futuros, o algoritmo deve sofrer melhorias.

Palavras-chave: Robôs Manipuladores, Puma 560, Cinemática Inversa, Sistemas Evolutivos, Algoritmos Genéticos.

Abstract

Solve the kinematics of a robot manipulator is a calculation of great importance in robotics, since these robots are used on a commercial scale and in various scientific research. However, this can be a very complex calculation. This paper proposes an alternative way to calculate the inverse kinematics for robot manipulators. As an experimental reference for this study, the robot Puma 560 was chosen and an evolutionary system based on Genetic Algorithms has been implemented in C. For testing the proposed method, the results were separated in two main categories: the end effector position calculation of the manipulator and the end position and orientation of the manipulator. The proposed method was analyzed for accuracy and processing time. With averages and standard deviation of 100 tests sequence to each configuration, it can be concluded that the method is very effective for examples where only the final position is considered, and that the algorithm is reasonable when the complexity of the problem increases greatly (considering the orientation of the manipulator). For this calculation, it was concluded that in future works, the algorithm must have improvements.

Keywords: Robot Manipulators, Puma 560, Inverse Kinematic, Evolutionary Systems, Genetic Algorithm.

Sumário

Lista de Ilustrações

Lista de Tabelas

1. Introdução

Robôs manipuladores vêm sendo utilizados em larga escala no ambiente científico. As aplicações variam desde um robô explorador que analisa rochas e as escava em outro planeta, até simples operações em linhas de produção como soldagem de peças mecânicas.

Tais aplicações têm em comum a necessidade de precisão de localização de um ponto ao final do braço robótico, onde se localiza o efetuador. Esse cálculo envolve todas as juntas de um robô manipulador, sejam elas prismáticas (promovem um deslocamento longitudinal em parte do braço robótico) ou de rotação (promovem um deslocamento angular em parte do braço robótico).

O cálculo da posição final do efetuador a partir dos valores dos ângulos de cada junta é conhecido como cinemática direta. Já o inverso, ou seja, a determinação de quais ângulos devem ser utilizados para que uma posição final conhecida seja alcançada, é chamado de cinemática inversa.

Para a cinemática direta, existe um método relativamente simples e eficaz, que resolve o problema de maneira direta e funciona para qualquer modelo de robô. Basta seguir um algoritmo e uma resposta única e verdadeira será encontrada.

Já para a cinemática inversa, o problema não é de tão fácil solução. Não existe uma regra geral para a resolução de todos os casos. Existem vários métodos de se encontrar a solução e, principalmente, existe mais de uma solução para o mesmo ponto. Essas características podem ser verificadas no robô Puma 560 (*Figura 1*).

Figura 1 *- Puma 560[2]*

O problema das várias soluções, pode ser facilmente observado, imaginando o manipulador como um braço humano. Para alcançar um objeto com a mão, existem várias configurações que se pode fazer com o braço. Essas configurações para o Puma 560 estão ilustradas na *Figura 2* a seguir.

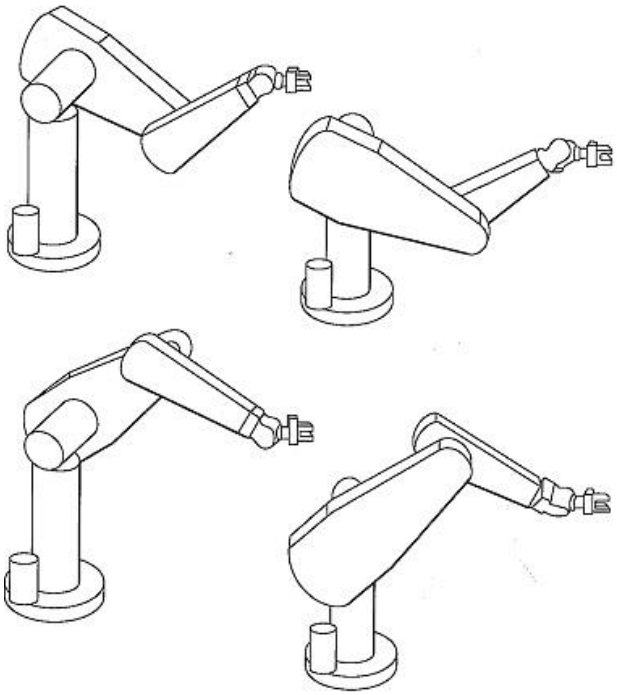

Figura 2 *- Configurações do Puma 560[2]*

Esse conjunto de fatores, faz com que a implementação da cinemática inversa seja demasiadamente custosa e complicada. Por isso, além de a solução analítica ou geométrica, que são mais comumente utilizadas, outros métodos de solução desse problema podem se tornar viáveis. Visto que o mais proibitivo nos métodos comuns do cálculo da cinemática inversa é muitas vezes a complexidade, a utilização de um cálculo através de métodos numéricos, utilizando-se algoritmos evolutivos torna-se justificada.

Existem diversos tipos de sistemas evolutivos, como algoritmos que simulam colônia de formigas, enxame de abelhas, dentre outros. Porém o mais difundido é o Algoritmo Genético.

O algoritmo genético tem como principais características resolver problemas complexos através de algoritmos relativamente simples, que se baseiam na Teoria da Evolução e na Genética criando modelos que se adaptam de maneira eficaz em várias áreas do conhecimento.

Em resumo, algoritmos genéticos buscam simular uma população de indivíduos, que evoluem através de processos como reprodução, mutação, seleção natural e outros conceitos evolucionistas, para que ao final os indivíduos resultantes sejam os mais adaptados possíveis ao ambiente em que vivem.

Analogamente, para a Cinemática Inversa, pode-se criar uma população, onde os indivíduos são possíveis ângulos para as juntas do Puma 560. Esses indivíduos sofrem as diversas adaptações e se tornam indivíduos que possuem os melhores ângulos que resultam em uma posição e orientação escolhidas para o efetuador. Resolvendo assim, o problema da Cinemática Inversa.

O presente trabalho busca, portanto, implementar um algoritmo genético para a resolução do problema da cinemática inversa do robô Puma 560. O algoritmo será desenvolvido na linguagem C e os resultados obtidos serão analisados e estudados, revelando assim, a viabilidade ou não desse modo de abordagem do problema da cinemática inversa.

O capítulo 2 trata da revisão bibliográfica, onde a fundamentação teórica necessária para o projeto, assim como o levantamento bibliográfico de pesquisas e trabalhos que serviram de base para a solução proposta nesse projeto serão apresentadas.

O capítulo 3 apresentará os métodos utilizados para a resolução do problema. É nesse capítulo que o algoritmo que foi utilizado será apresentado, detalhado e discutido.

O capítulo 4 discorre sobre a apresentação, análise e discussão dos resultados obtidos através do código do programa.

O capítulo 5 por fim, apresenta as considerações finais e conclusões do projeto, assim como propostas de trabalhos futuros.

2. Fundamentação Teórica

Neste capítulo serão introduzidos os principais conceitos utilizados na elaboração desse projeto. Para isso, serão analisadas as características mecânicas e funcionais do robô escolhido como referência, o equacionamento da cinemática direta para robôs manipuladores, o sistema de matrizes de rotação utilizado e uma visão geral de algoritmos genéticos. Por fim, será feito um levantamento de trabalhos que incentivaram essa pesquisa.

2.1. Puma 560

Criado pela empresa Unimation, o Puma 560 da série Puma 500 é utilizado tanto em aplicações comerciais como em ambientes de estudo como laboratórios de faculdade. Como já foi mencionado anteriormente, trata-se de um robô manipulador. O Puma possui seis juntas de rotação unidas por elos, e um efetuador ao final do braço.

Para os testes desse trabalho, resolveu-se modelar o Puma 560 com um segmento na extremidade do punho, gerando um novo ponto de posição final, e fazendo com que o cálculo da posição e da orientação final do robô seja mais interessante para demonstrar os objetivos propostos no trabalho.

Um esquema do robô em questão, com marcações de dimensões e sistemas de coordenadas está ilustrado a seguir na *Figura 3*.

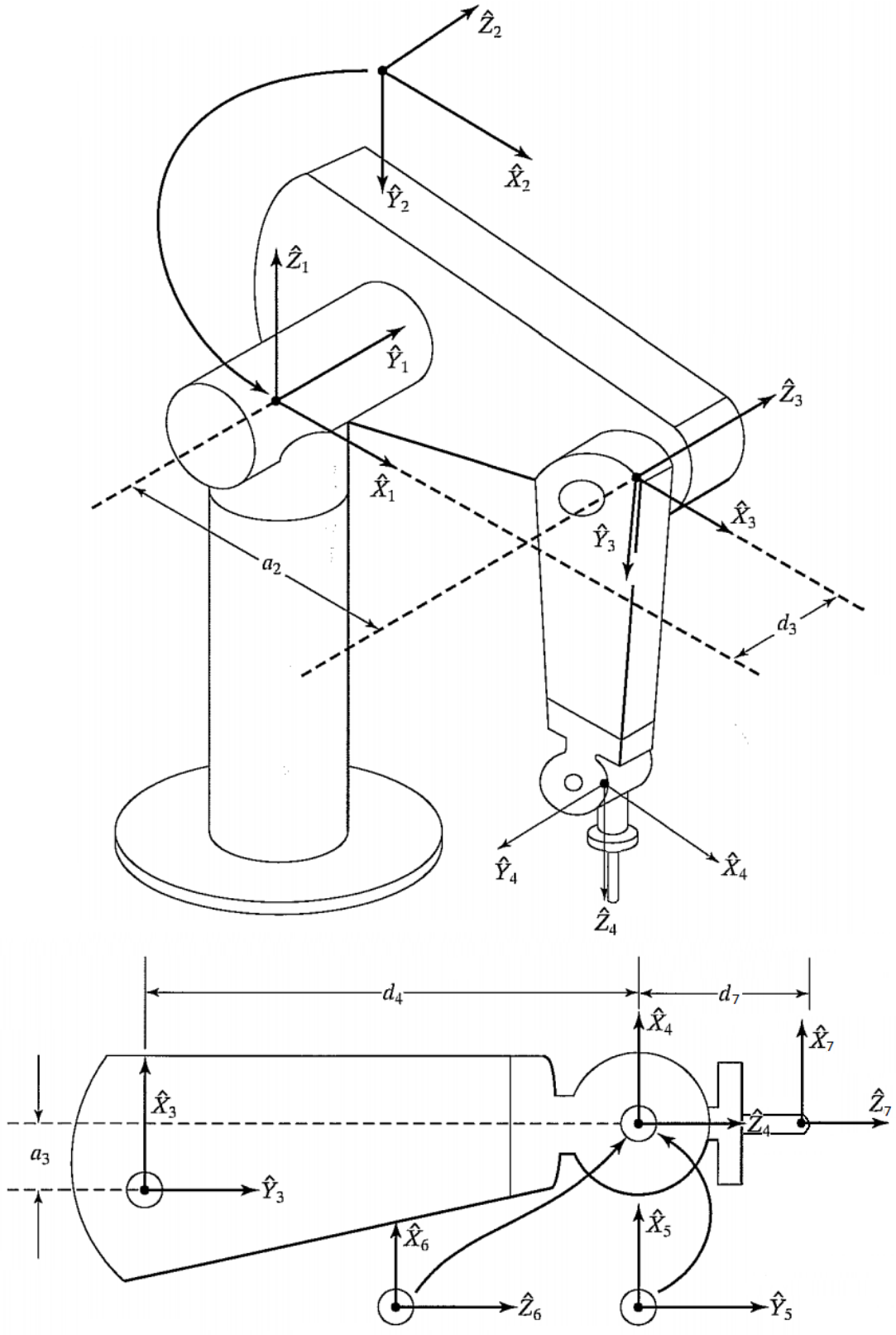

Figura 3 *- Esquema de Medidas do Puma560[2]*

Os tamanhos de cada elo foram retirados de medidas feitas em laboratório por outros trabalhos [4], sendo eles:

a_2 = 43,20cm, a_3 = 2,00cm, d_3 = 14,90cm, d_4 = 43,20cm e d_7 = 10,00cm.

A exceção é quanto a junta d_7 que teve sua medida criada para efeitos de estudo para esse trabalho.

Além disso, para efeitos de simulação, foi considerado o início do robô como a posição exata da primeira junta e todos os ângulos das juntas variando entre -180º e + 180º.

2.2. Cinemática Direta

Como já mencionado anteriormente neste trabalho, cinemática direta de um robô manipulador é o estudo da posição do seu efetuador em relação aos valores das suas juntas. Logo, através dela, também é possível definir qual será a posição e orientação do sistema de coordenadas do efetuador em relação ao sistema de coordenadas inicial, fixo na base (primeiro elo).

Como visto na figura anterior (*Figura 3*), além do sistema de coordenadas inicial e final, define-se também um sistema para cada uma das demais juntas. A definição de todos esses sistemas de coordenadas, foi realizada seguindo um método conhecido como Representação de Denavit-Hartenberg, através do qual, é possível determinar a posição e a orientação do sistema i em relação ao sistema anterior (i−1), pelo uso de matrizes homogêneas, relacionando a transformação entre estes sistemas.

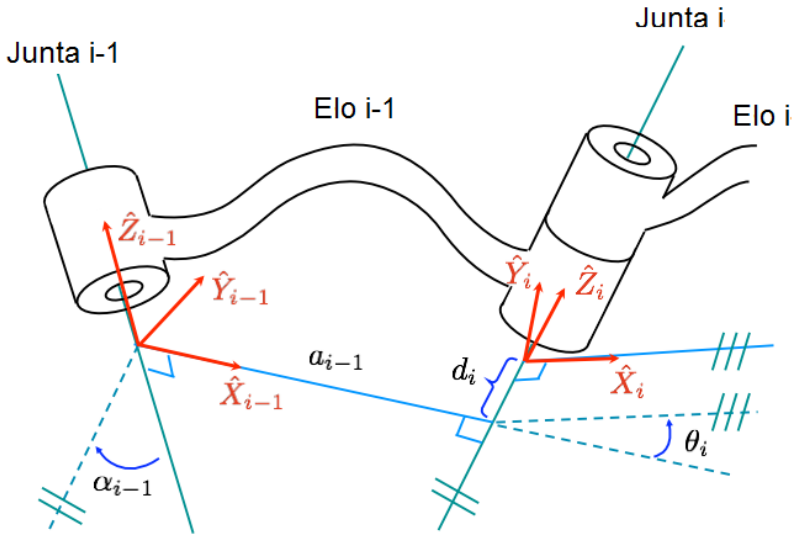

Figura 4 *- Representação de Denavit-Hartenberg[2]*

Dado um robô composto por juntas de rotação e elos entre elas, representado anteriormente na *Figura 4*, podem-se definir também os Parâmetros de Denavit-Hartenberg, sendo:

a_{i-1} o comprimento do elo (distância entre Z_i e Z_{i+1} ao longo de X_{i-1});
α_{i-1} a torção do elo (ângulo entre Z_i e Z_{i+1} em torno de X_{i-1});
d_i o deslocamento entre elos (distância entre X_{i-1} e X_i ao longo de Z_{i-1});
θ_i o ângulo da junta (ângulo entre X_{i-1} e X_i em torno de Z_{i-1}).

Conhecidas as notações e parâmetros, o sistema de coordenadas de cada junta, assim como os parâmetros de Denavit-Hartenberg, podem ser definidos através do método demonstrado na *Tabela 1* a seguir:

Tabela 1 *- Método de Denavit-Hartenberg*

- Numerar os elos a partir da base imóvel do manipulador. (elo 0);

- Desenhar linhas ao longo dos eixos de cada junta;

- Para o Primeiro Elo (i=0):
  $\{0\} = \{1\}$ para $\theta_1 = 0$ escolhido;
  $d_1 = 0$ constante;

- Para i=1 até i =n-1;
  - Posicionar a origem de Oi onde a perpendicular comum entre os eixos i e i+1 encontra com o eixo i. Se os eixos i e i+1 se cruzam, posicionar O_i nessa intersecção. Se i e i+ são paralelos, escolher O_i de forma conveniente;

  - Definir o eixo Z_i ao longo do eixo da junta i;

  - Definir o eixo X_i ao longo da perpendicular comum entre os eixos i e i+1. Se os eixos se interceptam, definir X_i normal ao plano contendo os dois eixos.

  - Definir Y_i de acordo com a regra da mão direita.

- Para o Último Elo (i = n):
  X_i se alinha com X_{i-1} para $\theta_n = 0$;
  Origem de {N} escolhida para que $d_n = 0$;

O resultado dos sistemas de coordenadas pode ser verificado na *Figura 3* já apresentada, e os parâmetros definidos pelo método podem ser encontrados a seguir na *Tabela 2*.

Tabela 2 *- Parâmetros de Denavit-Hartenberg*

i	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	0	0	0	θ_1
2	-90º	0	0	θ_2
3	0	a_2	d_3	θ_3
4	-90º	a_3	d_4	θ_4
5	90º	0	0	θ_5
6	-90º	0	0	θ_6
7	0	0	d_7	0

Dessa forma, a posição e a orientação do efetuador em relação à base são obtidas por uma composição de transformações homogêneas consecutivas, partindo-se do sistema da base para o sistema do efetuador.

Para calcular a matriz de transformação de um sistema O_{i-1} para o seguinte O_i basta seguir as seguintes transformações:

T_i^{i-1} = Rotação (X,α_{i-1}) × Translação(X, a_{i-1}) × Rotação(Z, θ_i) × Translação(Z, d_i)

Logo:

$$T_i^{i-1} = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & \cos(\alpha_{i-1}) & -\text{sen}(\alpha_{i-1}) & 0 \\ 0 & \text{sen}(\alpha_{i-1}) & \cos(\alpha_{i-1}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(\theta_i) & -\text{sen}(\theta_i) & 0 & 0 \\ \text{sen}(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} \cos(\theta_i) & -\text{sen}(\theta_i) & 0 & a_{i-1} \\ \text{sen}(\theta_i) * \cos(\alpha_{i-1}) & \cos(\theta_i) * \cos(\alpha_{i-1}) & -\text{sen}(\alpha_{i-1}) & -\text{sen}(\alpha_{i-1}) * d_i \\ \text{sen}(\theta_i) * \text{sen}(\alpha_{i-1}) & \cos(\theta_i) * \text{sen}(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1}) * d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dessa forma, para obter a matriz de transformação de T07, basta calcular:

$T_7^0 = T_1^0 \times T_2^1 \times T_3^2 \times T_4^3 \times T_5^4 \times T_6^5 \times T_7^6$;

O cálculo da matriz final de transformação entre o sistema inicial e final que conclui a cinemática direta (T_7^0) será realizado posteriormente no capítulo 3.

2.3. Matrizes de Rotação

Para uma fácil visualização da orientação final do efetuador, foi utilizado nesse trabalho, o sistema de rotação Roll-Pitch-Yaw. Esse sistema de rotação prevê que toda rotação pode ser representada pela combinação de três rotações consecutivas ao redor de cada eixo do sistemas de coordenadas inicial.

Considerando os ângulos roll para a rotação em torno de z, pitch para a rotação em torno de y e yaw para a rotação em torno de x, conforme a *Figura 5* a seguir.

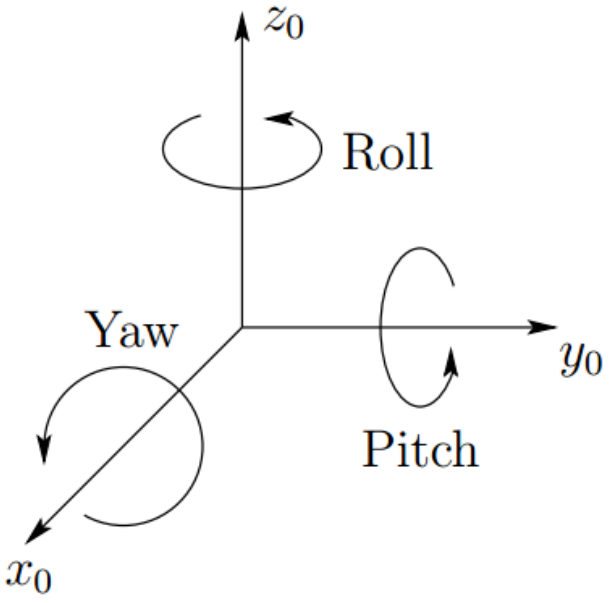

Figura 5 *- Rotação Roll-Pitch-Yaw[1]*

Sendo roll = Φ, pitch = θ e yaw = Ψ. A matriz final de Rotação pode ser apresentada como na equação [1] a seguir:

$R_{xyz} = R_{z,\Phi}, R_{y,\theta}, R_{x,\Psi}$

$$= \begin{bmatrix} cos(\Phi) & -sen(\Phi) & 0 \\ sen(\Phi) & cos(\Phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos(\theta) & 0 & sen(\theta) \\ 0 & 1 & 0 \\ -sen(\theta) & 0 & cos(\theta) \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\Psi) & -sen(\Psi) \\ 0 & sen(\Psi) & cos(\Psi) \end{bmatrix}$$

$$\begin{bmatrix} cos(\Phi)cos(\theta) & -sen(\Phi)cos(\Psi) + cos(\Phi)sen(\theta)sen(\Psi) & sen(\Phi)sen(\Psi) + cos(\Phi)sen(\theta)cos(\Psi) \\ sen(\Phi)cos(\theta) & cos(\Phi)cos(\Psi) + sen(\Phi)sen(\theta)sen(\Psi) & -cos(\Phi)sen(\Psi) + sen(\Phi)sen(\theta)cos(\Psi) \\ -sen(\theta) & cos(\theta)sen(\Psi) & cos(\theta)cos(\Psi) \end{bmatrix}$$

Esse será o sistema de rotação adotado nos cálculos posteriores, no capítulo 3.

2.4. Algoritmos Genéticos

Sistemas Evolutivos podem ser considerados como uma área de inteligência computacional que se baseia em empregar processos evolutivos sobre uma população de indivíduos que evoluem de acordo com processos inspirados pelas leis evolucionistas como o Darwinismo.

A definição dos diferentes tipos de Sistemas Evolutivos é muitas vezes semelhante e redundante, já que aconteceram por volta da mesma época, e sempre a partir de um mesmo princípio.

Nos Algoritmos Genéticos, além do princípio evolucionista, utiliza-se de operadores genéticos, tais como: reprodução, mutação e eliminação dos menos aptos, em algoritmos iterativos para alcançar indivíduos adaptados o suficientemente para resolver um problema.

A *Figura 7* a seguir, mostra um resumo de como a dinâmica do algoritmo genético usado nesse trabalho funciona. Em seguida, cada parte do fluxograma é explicada de maneira mais específica.

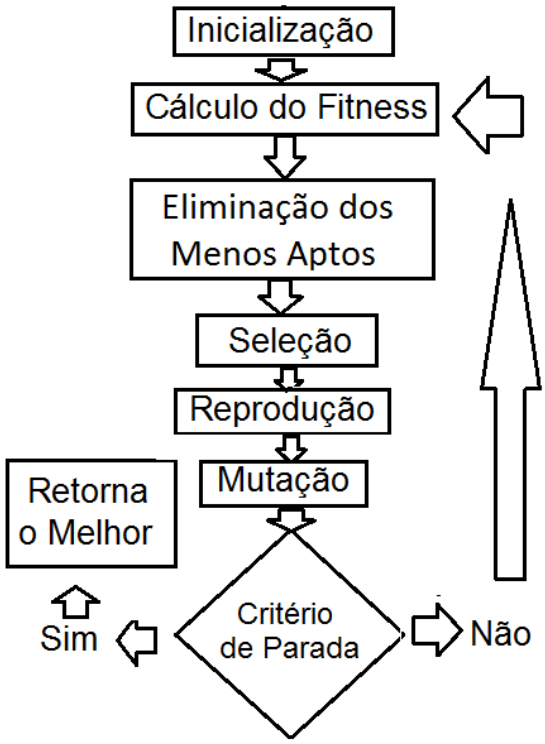

Figura 6 *- Fluxograma do Algoritmo Genético*

2.4.1.Inicialização

Como base de funcionamento para os Algoritmos Genéticos, uma população de tamanho fixo de indivíduos é criada. Essa população é criada de forma aleatória, com valores aleatórios para cada indivíduo. Existem ainda, outras maneiras de se inicializar a população, sendo possível levar em consideração algum conhecimento prévio na inicialização da mesma.

Cada um desses indivíduos criados têm uma estrutura composta por diversos parâmetros que serão detalhados no capítulo 3. Porém, para melhor exemplificação, os principais parâmetros são as possíveis soluções para um problema, que no contexto desse trabalho, são equivalentes aos valores de ângulos das juntas do robô manipulador.

Esses parâmetros podem ser comparados à essência daquele indivíduo, como um cromossomo. Ainda nessa comparação, cada ângulo representado pode ser comparado à um gene daquele cromossomo. A representação escolhida foi em valores numéricos para os ângulos em radianos. Porém, a mesma poderia ter sido feita com valores binário, hexadecimais, dentre outros.

2.4.3.Cálculo do Fitness

Uma função (no escopo desse trabalho, a cinemática direta) associa uma nota à determinado indivíduo. É assim que se pode mensurar o quão adaptado aquele indivíduo está. Essa nota é chamada de fitness, e nesse caso, foi definido como quanto menor, melhor.

A escolha da função de avaliação depende muito do problema e da maneira de representação dos indivíduos. Indivíduos representados por valores binários, por exemplo, requereriam uma função de avaliação completamente diferente.

O mais importante é que essa função consiga dar uma nota pra cada indivíduo, separando os mais adaptados dos menos adaptados. A maneira específica de como essa nota é calculada, será detalhada posteriormente no capítulo 3.

2.4.4.Eliminação dos Menos Aptos

A cada análise da população, uma análise de aptidão é realizada. Se após certo número pré-definido de gerações o fitness do melhor indivíduo não está se alterando, todos os demais indivíduos com exceção do melhor (elitismo) são eliminados, simulando um genocídio. Essa técnica ajuda a prevenir a convergência para mínimos e máximos locais.

Gerar novamente novos indivíduos faz com que um maior espaço de busca seja analisado, promovendo uma maior variabilidade nos indivíduos. O melhor indivíduo é salvo para não se perder a referência do melhor resultado encontrado até então.

2.4.5.Seleção e Reprodução

De maneira iterativa, as populações vão sendo substituídas por novas gerações. A seleção dos indivíduos é feita a partir da função de fitness já explicada. Essa função seleciona qual é o individuo daquela população com a melhor nota.

Selecionado o melhor indivíduo, existem vários métodos para se realizar a reprodução. Na clonagem, uma cópia idêntica dos indivíduos é transferida para a nova população. Já no cruzamento indivíduos são de alguma forma associado a outros.

Alguns métodos de cruzamento conhecidos são a Roleta Russa, onde pra cada pai um indivíduo é selecionado ao acaso para o cruzamento e o Torneio de N, onde para cada pai, n indivíduos são selecionados ao acaso, e o melhor deles é que cruzará com o pai para gerar um novo descendente.

Para esse projeto, foi utilizado o cruzamento do melhor, onde o melhor indivíduo da população cruza com todos os outros para gerar novos descendentes.

2.4.6.Mutação

Por fim, aleatoriamente, alguns indivíduos passam por um processo de mutação. Pequenas modificações na estrutura de cada indivíduo (como a soma de uma pequena quantia randômica no valor de algum ângulo) são feitas, promovendo novamente uma maior variabilidade, e causando uma diversificação interessante para que se possa encontrar um resultado novo. A mutação de uma parte da característica de um indivíduo pode ser considerada como um gene diferente em um cromossomo, que pode gerar uma característica nova, que ajude ou não para a sobrevivência do mesmo.

2.4.7.Critério de Parada

Como na natureza, o algoritmo genético está sempre em evolução. Portanto, deve-se criar um critério de parada, onde os resultados daquele momento sejam tomados como resposta. Nesse projeto, o critério adotado como parada foi a diferença entre o erro do melhor indivíduo com um erro mínimo pré-definido, garantindo uma solução satisfatória. Porém, caso isso não ocorra em até 1 segundo, a evolução também é encerrada.

2.5. Trabalhos Relacionados

Os trabalhos relacionados a seguir, buscam exemplificar trabalhos de diversas naturezas, voltados à resolução da cinemática inversa utilizando-se de algoritmos genéticos. Cada um deles, com uma abordagem própria, contribuiu para a análise e sedimentação de conceitos para que esse projeto fosse criado.

Dentre os diversos trabalhos relacionados pesquisados na área, seis trouxeram informações relevantes para a motivação desse projeto.

Uma das maneiras mais simples do uso de algoritmos genéticos para o cálculo da cinemática inversa foi encontrada em Scofano [7]. A motivação era resolver a cinemática inversa para o robô Braid, de 25 elos e com dois graus de liberdade por elo, em duas dimensões. Porém, por se tratar de um atuador binário, apenas um conjunto de soluções era possível. O posicionamento final era restringido há um conjunto de pontos no espaço.

Uma primeira evolução para esse trabalho foi encontrada em Nunes [6] e em Nunes, Rosado e Grandinetti [10]. Agora, o robô utilizado é o Robix RCS-6, de 3 graus de liberdade. Os resultados continuam em duas dimensões, porém agora se trata de um atuador contínuo (repostas em certo espaço continuo, não mais pontuais). Além disso, ambos os trabalhos realizam também um planejamento de uma trajetória descrita pelo robô. O algoritmo genético calcula os pontos iniciais e finais desejados, e os pontos intermediários são obtidos através do cálculo de uma trajetória cúbica. Nunes [6] ainda faz a comparação dos resultados implementados no MatLab e em linguagem C, concluindo que a última possui melhores resultados e rapidez de processamento.

Como sugestão para trabalhos futuros, Nunes [6] propõe a resolução para um sistema em três dimensões. Trabalho de Ramírez e Rubiano [11] resolve o problema da cinemática inversa para a posição final, com o robô Teachbot-01, de 3 graus de liberdade, porém dessa vez em três dimensões. Uma característica interessante desse trabalho é a forma de visualização de resultados implementada, possibilitando uma imagem próxima do que seria a configuração final do robô em questão.

Chapelle e Bidaud [9], resolvem o problema da posição para um outro robô que trabalha em ambientes de três dimensões, o Puma560. O robô possui 6 graus de liberdade, porém, só os 3 primeiros ângulos influenciam no cálculo da posição final do robô. O cálculo do posicionamento utilizando o algoritmo genético obteve sucesso.

Por fim, o trabalho de Santos, Lopes e Gebara [8] resolve a cinemática inversa para posição, usando algoritmos genéticos, para o robô Puma560, porém agora, adicionando um segmento ao final do braço robótico. Essa alteração faz com que os seis ângulos influenciem no

cálculo da posição, aumentando a complexidade do problema. Além disso, o planejamento de trajetória é feito, porém agora o algoritmo genético é usado para calcular todos os pontos da trajetória. Para a otimização dos resultados, um método de redução progressiva do espaço de busca foi utilizado.

Analisando todos esses trabalhos, esse trabalho tenta conciliar o aprendizado passado em cada um deles, para reproduzir a técnica de encontrar a cinemática inversa para o Puma 560, em três dimensões, com seis juntas de rotação. Considerou-se um segmento ao final do braço robótico, para a maior complexidade de resultados. Considerou-se ainda, não só a posição do efetuador, como sua orientação. O algoritmo será implementado na linguagem C. Como em outros trabalhos, o algoritmo genético será analisado quanto à viabilidade da solução encontrada.

.

3. Metodologia

A presente seção discutirá qual a metodologia utilizada para implementar cada parte do programa. Todas as considerações aqui têm como base os conceitos teóricos fundamentados no capítulo 2.

É importante ressaltar que foi adotado um método de resolução para o problema, separado em três modos:

No primeiro modo, apenas a posição final do efetuador é levada em consideração, um cálculo simples, direto, que desconsidera a orientação e gera um resultado que pode variar entre as várias configurações do puma 560 pode gerar para um mesmo ponto.

No segundo modo, assim como no primeiro, a posição final do efetuador é o objetivo da solução encontrada. Porém uma configuração de ângulos iniciais é tomada como base para a resolução do problema. Esse modo tem como objetivo aproximar a configuração da resposta encontrada a uma configuração anterior. Dessa forma, o manipulador não faria um movimento muito grande para mover entre duas posições próximas em uma trajetória.

No terceiro e último modo, além da posição final, a orientação final desejada também é calculada. Esse cálculo é feito levando em consideração os parâmetros de rotação roll, pitch e yaw.

3.1. Desenvolvimento em C

O desenvolvimento do programa que calcula a cinemática inversa através de algoritmos genéticos foi feito na linguagem C, no sistema operacional Windows.

Como não se trata de uma aplicação embarcada, e apenas uma simulação, o sistema operacional oferece compatibilidade para grande maioria dos softwares utilizados no desenvolvimento do programa.

A linguagem C, apesar de não possuir muitas ferramentas de visualização simples como possuem outras linguagens como MATLAB e Java, é bem otimizada para realizar todas as contas e é de fácil implementação.

Todo código foi feito a partir do início, sendo que as únicas bibliotecas prontas utilizadas foram as que são próprias da linguagem C.

3.2. Cinemática Direta

No Capítulo 2, pode-se observar quais são os parâmetros e configurações gerados pelo algoritmo de Denavit-Hartenberg. Observou-se também, como a transformação de um sistema para o outro pode ser obtida.

Dito isso, as multiplicações de matrizes foram realizadas e a matriz final encontrada foi:

Considerando:
$s_1 = \text{sen}(\theta_1)$ e $c_1 = \cos(\theta_1)$
$s_2 = \text{sen}(\theta_2)$ e $c_2 = \cos(\theta_2)$
$s_3 = \text{sen}(\theta_3)$ e $c_3 = \cos(\theta_3)$
$s_4 = \text{sen}(\theta_4)$ e $c_4 = \cos(\theta_4)$
$s_5 = \text{sen}(\theta_5)$ e $c_5 = \cos(\theta_5)$
$s_6 = \text{sen}(\theta_6)$ e $c_6 = \cos(\theta_6)$
$s_{23} = \text{sen}(\theta_2 + \theta_3)$; $c_{23} = \cos(\theta_2 + \theta_3)$

Tem-se que:

$T_7^0 = T_1^0 \times T_2^1 \times T_3^2 \times T_4^3 \times T_5^4 \times T_6^5 \times T_7^6$;

$T_7^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$, onde:

r_{11} = c_1*(c_{23}*((c_4*c_5*c_6) - (s_4*s_6)) - (s_{23}*s_5*c_6)) + s_1*((s_4*c_5*c_6) + (c_4*s_6))
r_{12} = s_1*(c_{23}*((c_4*c_5*c_6) - (s_4*s_6)) - (s_{23}*s_5*c_6)) - c_1*((s_4*c_5*c_6) + (c_4*s_6))
r_{13} = -s_{23}*((c_4*c_5*c_6) - (s_4*s_6)) - (c_{23}*s_5*c_6)
r_{21} = c_1*(c_{23}*(-(c_4*c_5*s_6) - (s_4*c_6)) + (s_{23}*s_5*s_6)) + s_1*((c_4*c_6) - (s_4*c_5*s_6))
r_{22} = s_1*(c_{23}*(-(c_4*c_5*s_6) - (s_4*c_6)) + (s_{23}*s_5*s_6)) - c_1*((c_4*c_6) - (s_4*c_5*s_6))
r_{23} = -s_{23}*(-(c_4*c_5*s_6) - (s_4*c_6)) + (c_{23}*s_5*s_6)
r_{31} = -c_1*((c_{23}*c_4*s_5) + (s_{23}*c_5)) - (s_1*s_4*s_5)
r_{32} = -s_1*((c_{23}*c_4*s_5) + (s_{23}*c_5)) + (c_1*s_4*s_5)
r_{33} = (s_{23}*c_4*s_5) - (c_{23}*c_5)
p_x = c_1*(c_{23}*(-c_4*s_5*d_7 + a_3)-s_{23}*(c_5*d_7 + d_4) + a_2*c_2) - s_1*(s_4*s_5*d_7 + d_3)
p_y = s_1*(c_{23}*(-c_4*s_5*d_7 + a_3) - s_{23}*(c_5*d_7 + d_4) + a_2*c_2) + c_1*(s_4*s_5*d_7 + d_3)
p_z = s_{23}*(c_4*s_5*d_7-a_3) - c_{23}*(c_5*d_7 + d_4) - a_2*s_2

Uma análise rápida, permite dizer que os valores p_x, p_y, p_z indicam quais são os valores do ponto [X,Y,Z] respectivamente, encontrados para os valores de ângulos de entrada da cinemática direta. Da mesma forma, os valores, r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33} indicam a matriz de rotação do efetuador em relação à base. Essa matriz de rotação pode ser comparada com a matriz de rotação gerada com os parâmetros de rotação Roll-Pitch-Yaw discutidos no capítulo anterior.

3.3. Estrutura dos Indivíduos e Criação da População

Para que o algoritmo genético tenha um bom funcionamento, é necessário definir uma boa estruturação de cada indivíduo.
No caso desse projeto, cada indivíduo é composto da seguinte forma:

Elementos principais, que definem todas as características do problema:

θ_1, θ_2, θ_3, θ_4, θ_5, θ_6 – Responsáveis por armazenar o valor do ângulo de cada junta do robô gerados por aquele indivíduo. Esse conjunto de informação que dá a identidade de cada indivíduo da população.

Elementos de consulta, que são resultado de processamentos realizados com os elementos principais:

r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, , r_{31}, r_{32}, r_{33} – Armazenam os valores de orientação encontrados pelo resultado do cálculo da cinemática direta para o conjunto de ângulos do indivíduo.

x, y, z – Armazenam os valores da posição final do efetuador calculados pela cinemática direta para o conjunto de ângulos do indivíduo.

Fitness – Responsável por armazenar a nota retornada da função de avaliação do algoritmo genético.

Erro – Responsável por armazenar o erro do indivíduo em relação ao fitness perfeito (zero).

De acordo com o método de inicialização escolhido, consiste em gerar valores aleatórios entre -180º e +180º para os ângulos θ_1, θ_2, θ_3, θ_4, θ_5, θ_6 de cada indivíduo.

3.4. Avaliação da População

Existem no programa três variáveis principais de avaliação.

A primeira (fitxyz) é responsável por avaliar a posição de cada indivíduo. É calculada somando-se as diferenças entre as posições desejadas e obtidas.

```
fitx = módulo(x-result[x])
fity = módulo(y-result[y])
fitz = módulo(z-result[z])
fitxyz = (fitx+fity+fitz);
```

A segunda (fitth) é responsável por avaliar a distância de cada ângulo de um indivíduo dos ângulos iniciais. É calculada somando-se as diferenças entre os ângulos iniciais e os do indivíduo.

```
fit1 = módulo(th1-thini1)
fit2 = módulo(th2-thini2)
fit3 = módulo(th3-thini3)
fit4 = módulo(th4-thini4)
fit5 = módulo(th5-thini5)
fit6 = módulo(th6-thini6)
fitth = (fit1+fit2+fit3+fit4+fit5+fit6)*180/pi;
```

Por fim, a terceira(fitr) é responsável por avaliar a distância entre a orientação obtida e a desejada. É calculada somando-se as diferenças entre os índices da matriz de rotação Roll-Pitch-Yaw desejada com os índices obtidos pela cinemática direta.

```
fit11 = módulo(rx1-result[3])
fit12 = módulo(rx2-result[4])
fit13 = módulo(rx3-result[5])
fit21 = módulo(ry1-result[6])
fit22 = módulo(ry2-result[7])
fit23 = módulo(ry3-result[8])
fit31 = módulo(rz1-result[9])
fit32 = módulo(rz2-result[10])
fit33 = módulo(rz3-result[11])
fitr = (fit11+fit12+fit13+fit21+fit22+fit23+fit31+fit32+fit33);
```

A partir daí, o cálculo do fitness é feito de acordo com o modo de execução que está sendo usado.

Se o modo é apenas a posição, fitness = fitxyz. Se o modo está buscando a posição levando em consideração os ângulos iniciais, fitness = 100*fitxyz + fitth. A multiplicação do valor de fitxyz por 100 serve para deixar os valores na mesma ordem de grandeza e, consequentemente, mesma importância na conta.

Se o terceiro modo for usado, onde a orientação também é considerada, o cálculo do fitness fica: fitness = 10*fitxyz + fitr. Novamente, a multiplicação do valor de fitxyz por 10

serve para deixar os valores na mesma ordem de grandeza e, consequentemente, mesma importância na conta.

3.5. Eliminação dos Menos Aptos

Depois de certo número de gerações em que o melhor indivíduo não evolui mais, uma rotina de elitismo é chamada. Essa rotina elimina todos os indivíduos com exceção do melhor. Porém dessa vez, os novos indivíduos gerados, são criados com valores aleatórios entre -360º e +360º. Posteriormente, os valores menores que -180º e maiores que +180º são normalizados para -180º e + 180º respectivamente.

Isso faz com que mais valores na borda sejam gerados, o que aumenta a velocidade de evolução em alguns casos, já que o método de cruzamento escolhido aproxima os valores para valores intermediários, pois é feito uma média de valores.

3.6. Cruzamento da População

Para o cruzamento dos indivíduos e consequente formação de uma nova geração o método do *cruzamento do melhor* é implementado da seguinte maneira:

Um indivíduo é selecionado aleatoriamente. O valor de cada ângulo desse indivíduo é somado ao valor do ângulo respectivo do melhor indivíduo e dividido por dois. A média de cada ângulo é então atribuída a um indivíduo da nova população.

O procedimento é repetido para cada indivíduo da nova população, com exceção do melhor, que é copiado na íntegra.

3.7. Mutação

A mutação ocorre para causar variabilidade genética. Para toda geração, é gerada uma probabilidade de mutação, um número aleatório entre 1 e 100. Se esse número estiver entre 1 e 5, 20% dos indivíduos sofreram mutação. Se estiver entre 5 e 10, 10% dos indivíduos sofreram mutação. Entre 10 e 30, 7% dos indivíduos sofrem mutações, entre 30 e 50, 3% dos indivíduos sofrem mutações, e entre 50 e 80, 1% dos indivíduos sofre mutação. Para números entre 80 e 100 não ocorre nenhuma mutação.

Após definido o número de mutações, sorteia-se quais indivíduos serão os que irão sofrer a mutação, e para cada um deles, sorteia-se ainda números entre -18º e + 18º para serem adicionados a cada ângulo.

Posteriormente, os valores menores que -180º e maiores que +180º são normalizados para -180º e + 180º respectivamente, para manter os resultados condizentes.

3.8. Critérios de Parada

Para cada modo de operação do algoritmo, um erro diferente é calculado. No primeiro e segundo modo, descritos no começo desse capítulo, o erro analisado é a variável fitxyz, descrita no item 3.5. Já para o último modo, o erro é a soma do fitxyz com o fitr, descrito no mesmo item.

Em todos esses casos, se o erro obtido for inferior a um erro pré-estipulado, o algoritmo para de evoluir e devolve o melhor indivíduo como resposta. Se por acaso o algoritmo levar mais de 1 segundo para evoluir, o critério de parada também é acionado.

3.9. Entrada e Saída de Dados

Para entrada de dados, exemplos de execução são definidos, onde devem ser configurados o modo de execução, e os valores necessários para cada modo.

O modo 1 requer apenas uma posição final (X,Y,Z).

No modo 2, além da posição final, devem ser configurados os ângulos iniciais thini1, thini2, thini3, thini4, thini5, thini6

No modo 3, devem ser configurados a posição final, e os parâmetros roll, pitch e yaw da orientação desejada em relação ao sistema base.

A saída também é dependente do modo de configuração. Em todos os casos, os ângulos de cada junta e a posição final do efetuador são exibidos. Porém, somente no modo 3, é exibida também a matriz de rotação Roll-Pitch-Yaw.

4. Resultados e Discussão

Ao presente capítulo, caberá a análise dos resultados gerados pelo programa, assim como a discussão da confiabilidade e viabilidade dos mesmos. Além de analisar os resultados gerados, é importante ressaltar quais foram as entradas usadas para colher tais resultados.

Para uma melhor compreensão, os resultados serão divididos entre os testes para a posição e os testes para posição e orientação do manipulador. A simulação foi divida em 6 testes.

Para cada um dos testes descritos a seguir, foram analisados 100 resultados. O melhor resultado de cada teste, assim como a média dos valores encontrados e o desvio padrão, são exibidos nas tabelas a seguir. Nos valores de erros das tabelas, o Erro da Posição, é a soma dos erros das posições x, y e z. O Erro nos Ângulos Iniciais é a soma da distância de todos os ângulos encontrados com o seu respectivo ângulo inicial. Por fim, o Erro de Orientação é a soma da diferença entre cada posição da matriz de rotação encontrada e da desejada.

4.1. Posição

4.1.1. Teste 1

Objetivo: Simples comprovação do modo 1 (só a posição final) para um ponto A qualquer.

Modo: 1.
Valores de Posição Desejados:
x = 3,40cm; y = 20,84cm; z = -20,67cm.
Possível solução que gera a posição e orientação desejadas:
θ_1= 30º, θ_2= -40º, θ_3= 60º, θ_4= 20º, θ_5= 25º, θ_6= 30º.

Tabela 3 *- Resultados do Teste 1*

Teste 1 – 100 valores	**Melhor**	**Média**	**Desvio Padrão**
Tempo(s)	0,024	0,145	0,248
Número de Iterações	340	1998	3408
Erro na Posição (cm)	0,01	0.13	0,29

Posição encontrada para o melhor:
x = 3,40cm; y = 20,84cm; z = -20,66cm.
Ângulos encontrados para o melhor:
θ_1= -66,00º; θ_2= -136,92º; θ_3= 120,02º; θ_4= -45,31º; θ_5= 27,83º; θ_6= -11,62º.

Analisando os resultados anteriores, pode-se concluir que são resultados muito satisfatórios, já que uma ótima precisão na localização da posição final foi garantida e o tempo de execução foi bem baixo.

4.1.2.Teste 2

Objetivo: Simples comprovação do modo 1 (só a posição final) para um ponto B qualquer próximo de A.

Modo: 1.
Valores de Posição Desejados:
x = 7,19cm; y = 24,15cm; z = -16,89cm.
Possível solução que gera a posição e orientação desejadas:
θ_1= 35º, θ_2= -45º, θ_3= 50º, θ_4= 5º, θ_5= 60º, θ_6= 90º.

Tabela 4 *- Resultados do Teste 2*

Teste 2 – 100 valores	**Melhor**	**Média**	**Desvio Padrão**
Tempo(s)	0,007	0,123	0,226
Número de Iterações	98	1702	3118
Erro na Posição (cm)	0,02	0,09	0,06

Posição encontrada para o melhor:
x = 7,19cm; y = 24,15cm; z = -16,88cm.
Ângulos encontrados para o melhor:
θ_1= -63,97º; θ_2= -140,37º; θ_3= 136,28º; θ_4= -12,77º; θ_5= -77,49º; θ_6= -157,86º.

Analisando os resultados anteriores, pode-se concluir que são resultados muito satisfatórios, já que uma ótima precisão na localização da posição final foi garantida e o tempo de execução foi bem baixo.

Além disso, pode-se perceber que os valores de ângulos encontrados são bem diferentes dos conhecidos na possível solução. Isso prova a multiplicidade de resultados do manipulador, já que um resultado satisfatório com outra configuração foi encontrado para a mesmo posição do efetuador.

Percebe-se no teste 2, em relação ao teste 1, que para sair do Ponto A (baseado nos ângulos: θ_1= 30º, θ_2= -40º, θ_3= 60º, θ_4= 20º, θ_5= 25º, θ_6= 30º) e caminhar até o ponto B próximo de A, ocorreu um grande deslocamento angular (θ_1= -63,97º; θ_2= -140,37º; θ_3= 136,28º; θ_4= -12,77º; θ_5= -77,49º; θ_6= -157,86º).

4.1.3.Teste 3

Objetivo: Comprovar a diferença entre os modos 1 (só a posição final) e 2 (posição final + ângulos iniciais) para o mesmo ponto B próximo de A.

Modo: 2
Valores de Posição Desejados:
x = 7,19cm; y = 24,15cm; z = -16,89cm.
Possível solução que gera a posição e orientação desejadas:
θ_1= 35º, θ_2= -45º, θ_3= 50º, θ_4= 5º, θ_5= 60º, θ_6= 90º.
Valor dos ângulos iniciais:
θ_1= 30º, θ_2= -40º, θ_3= 60º, θ_4= 20º, θ_5= 25º, θ_6= 30º.

Tabela 5 *- Resultados do Teste 3*

Teste 3 – 100 valores	**Melhor**	**Média**	**Desvio Padrão**
Tempo(s)	0,066	0,092	0,164
Número de Iterações	871	1207	2151
Erro na Posição (cm)	0,06	0,08	0,02
Erro nos Ângulos Iniciais (graus)	109,97	339,88	111,69

Posição encontrada para o melhor:
x = 7,17cm; y = 24,15cm; z = -16,85cm.
Ângulos encontrados para o melhor:
θ_1= 32,28º; θ_2= -55,61º; θ_3= 61,20º; θ_4= 36,28º; θ_5= 16,60º; θ_6= -36,20º.

Analisando os resultados anteriores, pode-se concluir que são resultados muito satisfatórios, já que uma ótima precisão na localização da posição final foi garantida e o tempo de execução foi bem baixo.

Além disso, pode-se perceber que os valores de ângulos encontrados dessa vez são bem mais próximos aos passados como ângulos iniciais. Isso prova que a aproximação implementada para ângulos iniciais funciona de maneira correta.

Diferentemente do que aconteceu até então com o Erro na Posição, o Erro nos Ângulos Iniciais apresentou uma média e desvio padrão alto. O que indica que o algoritmo nem sempre está chegando a uma solução satisfatória.

Percebe-se no teste 3, em relação ao teste 1, que para sair do Ponto A (baseado nos ângulos: θ_1= 30º, θ_2= -40º, θ_3= 60º, θ_4= 20º, θ_5= 25º, θ_6= 30º) e caminhar até o ponto B próximo de A, ocorreu um pequeno deslocamento angular (θ_1= 32,28º; θ_2= -55,61º; θ_3= 61,20º; θ_4= 36,28º; θ_5= 16,60º; θ_6= -36,20º). Isso demonstra a eficácia do modo 2 (teste 3) em relação ao modo 1(teste 2).

4.2. Posição + Orientação

4.2.1. Teste 4

Objetivo: Simples comprovação do funcionamento do modo 3 (posição final + orientação final).

Modo = 3.
Valores de Posição Desejados:
x = 45,20cm; y = 14,90cm; z = -53,20cm.
Possível solução que gera a posição e orientação desejadas:
$\theta_1 = \theta_2 = \theta_3 = \theta_4 = \theta_5 = \theta_6 = 0^o$.
Parâmetros de rotação escolhidos:
yaw = 0º, pitch = 180º e roll = 180º.
Matriz de rotação definida pelos parâmetros Roll, Pitch e Yaw: $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$.

Tabela 6 *- Resultados do Teste 4*

Teste 4 – 100 valores	Melhor	Média	Desvio Padrão
Tempo(s)	1,00	1,000	0,000
Número de Iterações	13121	13236	88
Erro na Posição (cm)	0,11	0,37	0,85
Erro na Orientação	0,16	3,97	2,43

Posição encontrada para o melhor:
x = 45,20cm; y = 14,90cm; z = -53,20cm.
Ângulos encontrados para o melhor:
$\theta_1 = -142{,}93^o$; $\theta_2 = -179{,}86^o$; $\theta_3 = -175{,}84^o$; $\theta_4 = -72{,}04^o$; $\theta_5 = -2{,}74^o$; $\theta_6 = -70{,}99^o$.
Matriz de rotação encontrada para o melhor: $\begin{bmatrix} 1{,}00 & 0{,}00 & 0{,}08 \\ 0{,}00 & -1{,}00 & 0{,}00 \\ 0{,}08 & 0{,}00 & -1{,}00 \end{bmatrix}$

Analisando os resultados anteriores, pode-se concluir que são resultados muito satisfatórios, já que uma ótima precisão na localização da posição final foi garantida, assim como uma ótima precisão na matriz de rotação. Porém, já é possível perceber que o algoritmo não convergiu em tempo suficiente, parando apenas no limite de 1 segundo. Apesar disso, o resultado ainda é muito satisfatório.

4.2.2. Teste 5

Objetivo: Comprovação do funcionamento do modo 3 (posição final + orientação final) para mesma posição com orientações diferentes (Orientação 1).

Modo = 3.
Valores de Posição Desejados:
x = 43,70cm; y = 11,55cm; z = 42,00cm.
Parâmetros de rotação escolhidos:
yaw = 0º, pitch = -90º e roll = 180º.
Matriz de rotação definida pelos parâmetros Roll, Pitch e Yaw: $\begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$.

Tabela 7 *- Resultados do Teste 5*

Teste 5 – 100 valores	**Melhor**	**Média**	**Desvio Padrão**
Tempo(s)	1,000	1,000	0,000
Número de Iterações	13577	13305	156
Erro na Posição (cm)	0,05	0,23	0,68
Erro na Orientação	0,43	4,88	1,80

Posição encontrada para o melhor:
x = 43,75cm; y = 11,55cm; z = 42,00cm.
Ângulos encontrados para o melhor:
θ_1= -3,84º; θ_2= -103,81º; θ_3= 15,25º; θ_4= -36,09º; θ_5= 4,34º; θ_6= 36,43º.
Matriz de rotação encontrada para o melhor: $\begin{bmatrix} 0,09 & -0,11 & 0,99 \\ -0,01 & -0,99 & -0,11 \\ 1,00 & 0,00 & -0,09 \end{bmatrix}$

Analisando os resultados anteriores, pode-se concluir que são resultados satisfatórios, já que uma ótima precisão na localização da posição final foi garantida, seguida de uma boa precisão na matriz de rotação. O tempo de execução novamente ultrapassou o limite de 1 segundo.

4.2.3. Teste 6

Objetivo: Comprovação do funcionamento do modo 3 (posição final + orientação final) para mesma posição com orientações diferentes (Orientação 2).

Modo = 3.
Valores de Posição Desejados:
x = 43,70cm; y = 11,55cm; z = 42,00cm.
Parâmetros de rotação escolhidos:
yaw = 0º, pitch = 0º e roll = 180º.
Matriz de rotação definida pelos parâmetros Roll, Pitch e Yaw: $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Tabela 8 *- Resultados do Teste 6*

Teste 6 – 100 valores	Melhor	Média	Desvio Padrão
Tempo(s)	1,000	1,000	0,000
Número de Iterações	13118	13230	64
Erro na Posição (cm)	0,02	0,36	1,22
Erro na Orientação	1,80	5,53	1,60

Posição encontrada para o melhor:
x = 43,70cm; y = 11,55cm; z = 41,98cm.
Ângulos encontrados para o melhor:
θ_1= -151,88º; θ_2= -87,46º; θ_3= 167,66º; θ_4= 152,54º; θ_5= -77,09º; θ_6= 27,97º.
Matriz de rotação encontrada para o melhor: $\begin{bmatrix} -0,99 & 0,02 & 0,11 \\ 0,04 & -0,82 & 0,57 \\ 0,11 & 0,57 & 081 \end{bmatrix}$

Analisando os resultados anteriores, pode-se concluir que são resultados razoáveis, já que uma ótima precisão na localização da posição final foi garantida, seguida de uma precisão ruim na matriz de rotação. O tempo de execução novamente ultrapassou o limite de 1 segundo.

A análise da média e do desvio padrão do Erro na Orientação revela que os valores de um modo geral estão distantes do objetivo de precisão.

5. Conclusão

O presente trabalho contou com uma variedade grande de testes, porém todos para provar um mesmo objetivo, o cálculo da cinemática inversa para o Puma560 utilizando-se algoritmos genéticos.

Os diferentes testes foram realizados, porque o trabalho tentou explorar várias áreas de utilização da cinemática inversa, tentando retirar todas as informações possíveis provenientes desse cálculo.

Ficou evidenciado que o algoritmo genético resolveu muito bem problemas simples como a definição da posição do efetuador.

Quando se trata de problemas mais complexos, como definição de posição e orientação do efetuador, o algoritmo genético obteve uma solução satisfatória, porém consideravelmente mais lenta e menos precisa.

Pela análise final dos resultados, pode-se concluir que o objetivo desse trabalho foi cumprido. A resolução do problema utilizando algoritmos genéticos é completamente viável e pode ser uma ótima alternativa para os métodos convencionais.

O tempo de execução, mesmo nos piores casos, não são proibitivos, e a precisão, mesmo nos piores casos, é mais do que suficiente para a maioria das aplicações. Portanto, o resultado é satisfatório, porém depende muito da aplicação.

5.1. Dificuldade e Limitações

As maiores dificuldades encontradas na execução desse trabalho, estão relacionadas com a convergência do algoritmo genético. Conforme o problema foi crescendo e a complexidade aumentando, maiores eram as dificuldades de fazer o algoritmo evoluir em um tempo hábil.

Percebeu-se que quanto mais valores são levados em consideração na função de fitness, maior é a chance de uma dada população inicial randômica ficar presa em mínimos e máximos locais.

Para amenizar esse problema, mais operadores genéticos foram criados e diferentes técnicas foram testadas. Isso gerou outra dificuldade, balancear inúmeras variáveis em busca de um bom resultado.

5.2. Contribuições

Dentre as contribuições desse trabalho, vale citar que foi um estudo completo do Puma560, considerando toda a sua mecânica e geometria, no espaço de três dimensões, com todas as juntas existentes. Logo, pode-se entender bem o funcionamento desse manipulador robótico.

Além disso, foi feito um bom estudo sobre técnicas e parâmetros de algoritmos genéticos, e dos cálculos de cinemáticas direta e inversa.

5.3. Trabalhos Futuros

Da análise dos dados obtidos, tem-se motivação para uma melhor estruturação de um novo algoritmo genético, com diferentes métodos de fitness e diferentes operadores genéticos, para que uma maior precisão e eficiência sejam alcançados.

Para complementar o estudo realizado, poderia ser adicionado ao código, uma técnica de cálculo de trajetória, não se restringindo mais há apenas pontos isolados. Após isso, existe também a possibilidade de se considerar o problema do desvio de obstáculos no planejamento da trajetória.

Referências

[1] Mark W. Spong; Seth Hutchinson; M. Vidyasagar; "Robot Modeling and Control", John Wiley, 2006.

[2] John J. Craig; "Introduction to Robotics: Mechanics and Control", 3rd edition, Pearson-Prentice Hall, 2005.

[3] Evolutionary Computation: A Unified Approach, Kenneth A. De Jong, MIT Press, Cambridge, 2006.

[4] Peter I. Corke, Brian Armst rong-Hdouvry, A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot.

[5] http://www.antenen.com/htdocs/downloads/files/files_dl/puma560.pdf, Acessado em 15/11/12, *Datasheet* do robô Puma560.

[6] Nunes, Luiz Eduardo Nicolini do Patrocínio, Geração e otimização de trajetórias de um manipulador robótico utilizando algoritmos genéticos, Universidade Estadual Paulista, Guaratinguetá, 2007.

[7] Felipe S. Scofano, Obtenção da Cinemática Inversa de Robôs 2D Binários Hiper-Redundantes por Algoritmos Genéticos, Pontifícia Universidade Católica do Rio de Janeiro.

[8] Alfranci Freitas Santos, Heitor Silvério Lopes, Munif Gebara Junior, Cinemática Inversa de Trajetórias de Manipuladores Robóticos Redundantes utilizando Algoritmos Genéticos com Redução progressiva do espaço de busca, Centro Federal de Educação Tecnológica do Paraná, 2005.

[9] F. Chapelle, P., Bidaud, A, Closed Form for Inverse Kinematics Approximation of General 6R Manipulators using Genetic Programming, Université Paris, 2001.

[10] Luiz Eduardo Nicolini do Patrocínio Nunes, Victor Orlando Gamarra Rosado, Francisco José Grandinetti, Aplicação de algoritmos genéticos e polinômios cúbicos na geração de trajetórias de um manipulador robótico, Universidade de Taubaté.

[11] J. Ramírez A., A. Rubiano F, Optimization of Inverse Kinematics of a 3R Robotic Manipulator using Genetic Algorithms, World Academy of Science, Engineering and Technology, 2011.

Apêndice A – Cinemática Inversa com Algoritmo Genético em C

A seguir encontra-se o código completo utilizado no presente projeto, desenvolvido em C, para calcular a cinemática inversa utilizando algoritmos genéticos.

```
/*Libraries*/
#include "stdafx.h"
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include "time.h"
#include <iostream>
#include <windows.h>
#include <fstream>
#include <sys/timeb.h>

/*Definições Matemáticas*/
#define pi acos(-1.0)

/*Dimensões do Robô em cm - A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot*/
double a2 = 43.2;
double a3 =  2.0;
double d3 = 14.9;
double d4 = 43.2;
double d7 = 10.0;//Valor fictício, pois o braço foi inventado apenas para simulação.

/*Valores para a Posição Desejada*/
double x, y, z;
/*Ângulos Iniciais*/
double thini1; double thini2; double thini3; double thini4; double thini5; double thini6;
/*Valores para a Orientação Desejada*/
double yaw, pitch, roll;
double sroll, croll, spitch, cpitch, syaw, cyaw;//Seno e Cosseno dos ângulos roll, pitch e yaw.
double rx1,rx2,rx3,ry1,ry2,ry3,rz1,rz2,rz3;//Valores de translação.
/*Modo de Operação*/
double mode;//0 - Posição, 1 - Posição + Ângulos Anteriores, 2 - Posição + Orientação.
double samples = 100;//Número de amostras para cada teste.

/*Variaveis usadas para calculos estatisticos*/
double media_tempo, desvpadr_tempo;
double media_interacoes, desvpadr_interacoes;
double media_erropos, desvpadr_erropos;
double media_erroant, desvpadr_erroant;
double media_erroori, desvpadr_erroori;

/*Pârametros do Algoritmo Evolutivo*/
#define pop        100//Tamanho da população.
double  acerto    =   0.1;//Erro máximo permitido.
double  genoTaxa  = 200;//Diferença entre bests para incrementar o genocidio.
double  genocidio =  10;//Número de genoTaxas necessárias para ocorrer o genocidio.

/*Estrutura de cada indivíduo*/
struct individuo{
            double th1, th2, th3, th4, th5, th6;
            double r11, r12, r13, r21, r22, r23, r31, r32, r33;
            double x,y,z;
            double fitness;
            double erropos, erroant, erroori;
            double erro;
};

/*Estrutura das Respostas*/
struct respostas{
            double x,y,z;
```

```
		double th1, th2, th3, th4, th5, th6;
		double r11, r12, r13, r21, r22, r23, r31, r32, r33;
		double fitness;
		double erropos, erroant, erroori;
		double tempo;
		int geracoes;
};



/*Definindo as Populações*/
struct individuo parents[pop];//População de Pais
struct individuo childrens[pop];//População de Filhos
struct respostas resposta[100];//População de Respostas

/*Função que calcula a posição x,y,z pela cinemática direta*/
double * cinematicaDireta(int num_indiv)
{
		double s1, s2, s3, s4, s5, s6;//Seno dos ângulos.
		double c1, c2, c3, c4, c5, c6;//Cosseno dos ângulos.
		double s23,  c23;//Seno e Cosseno da soma dos ângulos.
		double r11, r21, r31, r12, r22, r32, r13, r23, r33;//Valores de rotação.
		double px, py, pz;//Valores de translação.

		double th1 = parents[num_indiv].th1;
		double th2 = parents[num_indiv].th2;
		double th3 = parents[num_indiv].th3;
		double th4 = parents[num_indiv].th4;
		double th5 = parents[num_indiv].th5;
		double th6 = parents[num_indiv].th6;

		s1 = sin(th1); s2 = sin(th2); s3 = sin(th3); s4 = sin(th4); s5 = sin(th5); s6 = sin(th6);//Definição do seno dos
ângulos.
		c1 = cos(th1); c2 = cos(th2); c3 = cos(th3); c4 = cos(th4); c5 = cos(th5); c6 = cos(th6);//Definição do
cosseno dos ângulos.
		s23 = sin(th2+th3); c23 = cos(th2+th3);//Definição do seno e do cosseno das somas dos ângulos.

		r11 = c1*(c23*((c4*c5*c6)-(s4*s6)) - (s23*s5*c6))+s1*((s4*c5*c6)+(c4*s6));//Linha 1, coluna 1, da
matriz de transformação.
		r21 = s1*(c23*((c4*c5*c6)-(s4*s6)) - (s23*s5*c6))-c1*((s4*c5*c6)+(c4*s6));//Linha 2, coluna 1, da matriz
de transformação.
		r31 = -s23*((c4*c5*c6)-(s4*s6))-(c23*s5*c6);//Linha 3, coluna 1, da matriz de transformação.

		r12 = c1*(c23*(-(c4*c5*s6)-(s4*c6))+(s23*s5*s6))+s1*((c4*c6)-(s4*c5*s6));//Linha 1, coluna 2, da matriz
de transformação.
		r22 = s1*(c23*(-(c4*c5*s6)-(s4*c6))+(s23*s5*s6))-c1*((c4*c6)-(s4*c5*s6));//Linha 2, coluna 2, da matriz
de transformação.
		r32 = -s23*(-(c4*c5*s6)-(s4*c6))+(c23*s5*s6);//Linha 3, coluna 2, da matriz de transformação.

		r13 = -c1*((c23*c4*s5)+(s23*c5))-(s1*s4*s5);//Linha 1, coluna 3, da matriz de transformação.
		r23 = -s1*((c23*c4*s5)+(s23*c5))+(c1*s4*s5);//Linha 2, coluna 3, da matriz de transformação.
		r33 = (s23*c4*s5)-(c23*c5);//Linha 3, coluna 3, da matriz de transformação.

		px = c1*(c23*(-c4*s5*d7+a3)-s23*(c5*d7+d4)+a2*c2)-s1*(s4*s5*d7+d3);//Linha 1, coluna 4, da matriz de
transformação.
		py = s1*(c23*(-c4*s5*d7+a3)-s23*(c5*d7+d4)+a2*c2)+c1*(s4*s5*d7+d3);//Linha 2, coluna 4, da matriz
de transformação.
		pz = s23*(c4*s5*d7-a3)-c23*(c5*d7+d4)-a2*s2;//Linha 3, coluna 4, da matriz de transformação.

		parents[num_indiv].r11 = r11;
		parents[num_indiv].r12 = r12;
		parents[num_indiv].r13 = r13;
		parents[num_indiv].r21 = r21;
		parents[num_indiv].r22 = r22;
		parents[num_indiv].r23 = r23;
		parents[num_indiv].r31 = r31;
		parents[num_indiv].r32 = r32;
```

```
		parents[num_indiv].r33 = r33;
		parents[num_indiv].x = px;
		parents[num_indiv].y = py;
		parents[num_indiv].z = pz;

		double result[12] = {px,py,pz, r11,r12,r13,r21,r22,r23,r31,r32,r33};//Reultados da translação e da rotação.

		return result;
}

/*Cálculo da função de Fitness*/
double nota(int num_indiv)
{
		double *result = cinematicaDireta(num_indiv);//Recebe os valores da cinemática direta.

		/*Atribui os valores calculados para o individuo*/
		double th1 = parents[num_indiv].th1;
		double th2 = parents[num_indiv].th2;
		double th3 = parents[num_indiv].th3;
		double th4 = parents[num_indiv].th4;
		double th5 = parents[num_indiv].th5;
		double th6 = parents[num_indiv].th6;

		/*Calcula o fitness para o individuo*/
		double fitness;//Fitness total.
		double fitx, fity, fitz;//Fitness da posição.
		double fit1, fit2, fit3, fit4, fit5, fit6;//Fitness dos ângulos anteriores.
		double fit11, fit12, fit13, fit21, fit22, fit23, fit31, fit32, fit33;//Fitness dos ângulos de rotação.
		double fitxyz = -1, fitr = -1, fitth = -1;//Fitness de cada modo.

		/*Posição*/
		fitx = x-result[0]; if(fitx<0) fitx = fitx*(-1);
		fity = y-result[1]; if(fity<0) fity = fity*(-1);
  fitz = z-result[2]; if(fitz<0) fitz = fitz*(-1);

		fitxyz = (fitx+fity+fitz);
		fitness = fitxyz;
		parents[num_indiv].erro = fitxyz;

		/*Ângulos Iniciais*/
		if(mode == 1){
			fit1 = th1-thini1; if(fit1<0) fit1 = fit1*(-1);
			fit2 = th2-thini2; if(fit2<0) fit2 = fit2*(-1);
			fit3 = th3-thini3; if(fit3<0) fit3 = fit3*(-1);
			fit4 = th4-thini4; if(fit4<0) fit4 = fit4*(-1);
			fit5 = th5-thini5; if(fit5<0) fit5 = fit5*(-1);
			fit6 = th6-thini6; if(fit6<0) fit6 = fit6*(-1);

			fitth = (fit1+fit2+fit3+fit4+fit5+fit6)*180/pi;
			fitness = 100*fitxyz + fitth;
			parents[num_indiv].erro = fitxyz;
		}

		/*Orientação*/
		if(mode == 2){
			fit11 = rx1-result[3];  if(fit11<0) fit11 = fit11*(-1);
			fit12 = rx2-result[4];  if(fit12<0) fit12 = fit12*(-1);
			fit13 = rx3-result[5];  if(fit13<0) fit13 = fit13*(-1);
			fit21 = ry1-result[6];  if(fit21<0) fit21 = fit21*(-1);
			fit22 = ry2-result[7];  if(fit22<0) fit22 = fit22*(-1);
			fit23 = ry3-result[8];  if(fit23<0) fit23 = fit23*(-1);
			fit31 = rz1-result[9];  if(fit31<0) fit31 = fit31*(-1);
			fit32 = rz2-result[10]; if(fit32<0) fit32 = fit32*(-1);
			fit33 = rz3-result[11]; if(fit33<0) fit33 = fit33*(-1);

			fitr = (fit11+fit12+fit13+fit21+fit22+fit23+fit31+fit32+fit33);
			fitness = fitxyz + fitr;
```

```
                parents[num_indiv].erro =  10*fitxyz + fitr;
        }

        parents[num_indiv].erropos = fitxyz;
        parents[num_indiv].erroant = fitth;
        parents[num_indiv].erroori = fitr;

        return fitness;
}

int main()
{
   /*Variáveis para contar tempo*/
        struct timeb ini, fim;

        srand(time(NULL));//Gerando numeros aleatorios.

        int best = 0;//Posição do melhor.
        int kill = 0;//Constante de genocídio.
        double bestant = 10000000;//Melhor da geração anterior, valor alto para não entrar a primeira vez no kill++.
        int parent1 = 0, parent2 = 0;//Escolha dos dois individuos para o cruzamento.
        int mut = 0, muta = 0, mutb = 0;//Números para armazenar os valores de escolha da mutação.
        double plus = 0.0;//Número a ser somado para que ocorra a mutação.
        int geracoes_cont = 0;//Número de gerações.
        int best_resposta = -1;//Melhor resposta da Bateria de Respostas.
        double resposta_ant;//Resposta anterior da Bateria de Respostas.

        /*Bateria de Testes*/
        for(int teste=1;teste<=6;teste++){
                /*Bateria de Respostas*/
                for(int tries=0;tries<samples;tries++){

                        ftime(&ini);//Reseta a contagem de Tempo.
                        geracoes_cont = 0;//Reseta a contagem de Iterações.

                        //Teste só com a Posição final - Aleatória.
                        if(teste == 1){
                                mode = 0;
                                x = 3.40; y = 20.84; z = -20.67;

                        //Teste só com a Posição final - Valor para todos os ângulos zeros.
                        }else if(teste == 2){
                                mode = 0;
                                x = 7.19; y = 24.15; z = -16.89;

                        //Teste com a Posição Final + Ângulos Anteriores - Valor para todos os ângulos zeros.
                        }else if(teste == 3){
                                mode = 1;
                                x = 7.19; y = 24.15; z = -16.89;
                                thini1 = 30*pi/180; thini2 = -40*pi/180; thini3 = 60*pi/180; thini4 =
20*pi/180; thini5 = 25*pi/180; thini6 = 30*pi/180;

                        //Teste com a Posição + Orientação - Valor para todos os ângulos zeros, sem rotação.
                        }else if(teste == 4){
                                mode = 2;
                                x = 45.20, y = 14.90, z = -53.20;
                                yaw = 0*pi/180, pitch = 180*pi/180, roll = 180*pi/180;

                                sroll = sin(roll); croll = cos(roll);
                                spitch = sin(pitch); cpitch = cos(pitch);
                                syaw = sin(yaw); cyaw = cos(yaw);
                                rx1 = croll*cpitch;
                                rx2 = -sroll*cyaw+croll*spitch*syaw;
                                rx3 = sroll*syaw+croll*spitch*cyaw;
                                ry1 = sroll*cpitch;
                                ry2 = croll*cyaw+sroll*spitch*syaw;
                                ry3 = -croll*syaw+sroll*spitch*cyaw;
```

```
                rz1 = -spitch;
                rz2 = cpitch*syaw;
                rz3 = cpitch*cyaw;

        //Teste com a Posição + Orientação - 1a. Orientação Escolhida
        }else if(teste == 5){
                mode = 2;
                //d7 = 0; a3 = 0;//Mudando os parametros do robô para uma melhor
visualização.
                x = 43.70, y = 11.55, z = 42.00;
                yaw = 0*pi/180, pitch = -90*pi/180, roll = 180*pi/180;

                sroll = sin(roll); croll = cos(roll);
                spitch = sin(pitch); cpitch = cos(pitch);
                syaw = sin(yaw); cyaw = cos(yaw);
                rx1 = croll*cpitch;
                rx2 = -sroll*cyaw+croll*spitch*syaw;
                rx3 = sroll*syaw+croll*spitch*cyaw;
                ry1 = sroll*cpitch;
                ry2 = croll*cyaw+sroll*spitch*syaw;
                ry3 = -croll*syaw+sroll*spitch*cyaw;
                rz1 = -spitch;
                rz2 = cpitch*syaw;
                rz3 = cpitch*cyaw;

        //Teste com a Posição + Orientação - 2a. Orientação Escolhida
        }else if(teste == 6){
                mode = 2;
                //d7 = 0; a3 = 0;//Mudando os parametros do robô para uma melhor
visualização.
                x = 43.70, y = 11.55, z = 42.00;
                yaw = 0*pi/180, pitch = 0*pi/180, roll = 180*pi/180;

                sroll = sin(roll); croll = cos(roll);
                spitch = sin(pitch); cpitch = cos(pitch);
                syaw = sin(yaw); cyaw = cos(yaw);
                rx1 = croll*cpitch;
                rx2 = -sroll*cyaw+croll*spitch*syaw;
                rx3 = sroll*syaw+croll*spitch*cyaw;
                ry1 = sroll*cpitch;
                ry2 = croll*cyaw+sroll*spitch*syaw;
                ry3 = -croll*syaw+sroll*spitch*cyaw;
                rz1 = -spitch;
                rz2 = cpitch*syaw;
                rz3 = cpitch*cyaw;
        }

        //Inicialização da população original randômicamente.
        for(int i=0; i<pop; i++){
                parents[i].th1 = rand()%36000;
                parents[i].th1 = ((parents[i].th1/100.0f)-180)*pi/180;parents[i].th1;
                parents[i].th2 = rand()%36000;
                parents[i].th2 = ((parents[i].th2/100.0f)-180)*pi/180;parents[i].th2;
                parents[i].th3 = rand()%36000;
                parents[i].th3 = ((parents[i].th3/100.0f)-180)*pi/180;parents[i].th3;
                parents[i].th4 = rand()%36000;
                parents[i].th4 = ((parents[i].th4/100.0f)-180)*pi/180;parents[i].th4;
                parents[i].th5 = rand()%36000;
                parents[i].th5 = ((parents[i].th5/100.0f)-180)*pi/180;parents[i].th5;
                parents[i].th6 = rand()%36000;
                parents[i].th6 = ((parents[i].th6/100.0f)-180)*pi/180;parents[i].th6;
        }

        /*Avaliação*/
        while (1){
                /*Cálculo das notas de cada individuo*/
                for(int i=0; i<pop; i++){
```

```
                parents[i].fitness = nota(i);
        }

        /*Cálculo do melhor individuo*/
        for(int i=0; i<pop; i++){
                if(parents[i].fitness <= parents[best].fitness){
                        best = i;
                }
        }

        /*Aumentando a constante de genocídio, caso a nota do melhor continue a
mesma*/

        if((bestant - parents[best].fitness) <= genoTaxa){
                kill++;
        }

        /*Genocidio. Recria todos os individuos, com exceção do melhor*/
        if(kill >= genocidio){
                for(int i=0; i<pop; i++){
                        if(i != best){
                                parents[i].th1 = rand()%72000;
                                parents[i].th1 = ((parents[i].th1/100.0f)-
360)*pi/180;
                                parents[i].th2 = rand()%72000;
                                parents[i].th2 = ((parents[i].th2/100.0f)-
360)*pi/180;
                                parents[i].th3 = rand()%72000;
                                parents[i].th3 = ((parents[i].th3/100.0f)-
360)*pi/180;
                                parents[i].th4 = rand()%72000;
                                parents[i].th4 = ((parents[i].th4/100.0f)-
360)*pi/180;
                                parents[i].th5 = rand()%72000;
                                parents[i].th5 = ((parents[i].th5/100.0f)-
360)*pi/180;
                                parents[i].th6 = rand()%72000;
                                parents[i].th6 = ((parents[i].th6/100.0f)-
360)*pi/180;
                        }
                        /*Regularizando Individuos*/
                        if(parents[i].th1 >  180*pi/180) parents[i].th1 =
180*pi/180;
                        else if(parents[i].th1 < -180*pi/180) parents[i].th1 = -
180*pi/180;
                        if(parents[i].th2 >  180*pi/180) parents[i].th2 =
180*pi/180;
                        else if(parents[i].th2 < -180*pi/180) parents[i].th2 = -
180*pi/180;
                        if(parents[i].th3 >  180*pi/180) parents[i].th3 =
180*pi/180;
                        else if(parents[i].th3 < -180*pi/180) parents[i].th3 = -
180*pi/180;
                        if(parents[i].th4 >  180*pi/180) parents[i].th4 =
180*pi/180;
                        else if(parents[i].th4 < -180*pi/180) parents[i].th4 = -
180*pi/180;
                        if(parents[i].th5 >  180*pi/180) parents[i].th5 =
180*pi/180;
                        else if(parents[i].th5 < -180*pi/180) parents[i].th5 = -
180*pi/180;
                        if(parents[i].th6 >  180*pi/180) parents[i].th6 =
180*pi/180;
                        else if(parents[i].th6 < -180*pi/180) parents[i].th6 = -
180*pi/180;
                }
                kill = 0;//Reseta a contagem de genocidio.
        }
```

```
/*Seleção*/
for(int i=0; i < pop; i++){
	parent1 = best;
	parent2 = rand()%pop;//Sorteia um indivíduo.

	/*Com exceção do melhor, cruza*/
	if (i == best)
		childrens[i] = parents[best];
	else{
		childrens[i].th1 =
(parents[parent1].th1+parents[parent2].th1)/2;
		childrens[i].th2 =
(parents[parent1].th2+parents[parent2].th2)/2;
		childrens[i].th3 =
(parents[parent1].th3+parents[parent2].th3)/2;
		childrens[i].th4 =
(parents[parent1].th4+parents[parent2].th4)/2;
		childrens[i].th5 =
(parents[parent1].th5+parents[parent2].th5)/2;
		childrens[i].th6 =
(parents[parent1].th6+parents[parent2].th6)/2;
	}

	/*Nova geração*/
	parents[i] = childrens[i];
}

/*Mutação*/
muta = rand()%100;//Sorteia um número, dependendo do resultado decide
quantas vezes mutar.
if(muta>80 && muta>=50)
	mut = 1;
else if(muta<50 && muta>=30)
	mut = 3;
else if(muta<30 && muta>=10)
	mut = 7;
else if(muta<10 && muta>=5)
	mut = 10;
else if(muta<5 && muta>=1)
	mut = 20;

/*Muta o numero de vezes escolhidas*/
for(int i=0; i< mut; i++){
	/*Escolhe alguém aleatoriamente para mutar, com excecão do
melhor*/
	mutb = rand()%pop;
	if(mutb != best){
		plus = rand()%3600;
		plus = (plus/100.0f)-18.00;
		parents[mutb].th1 = parents[mutb].th1 + plus;
		plus = rand()%3600;
		plus = (plus/100.0f)-18.00;
		parents[mutb].th2 = parents[mutb].th2 + plus;
		plus = rand()%3600;
		plus = (plus/100.0f)-18.00;
		parents[mutb].th3 = parents[mutb].th3 + plus;
		plus = rand()%3600;
		plus = (plus/100.0f)-18.00;
		parents[mutb].th4 = parents[mutb].th4 + plus;
		plus = rand()%3600;
		plus = (plus/100.0f)-18.00;
		parents[mutb].th5 = parents[mutb].th5 + plus;
		plus = rand()%3600;
		plus = (plus/100.0f)-18.00;
		parents[mutb].th6 = parents[mutb].th6 + plus;
	}
```

```
                                        /*Regularizando Individuos*/
                                        if(parents[mutb].th1 >  180*pi/180) parents[mutb].th1 =
180*pi/180;
                                        else if(parents[mutb].th1 < -180*pi/180) parents[mutb].th1 = -
180*pi/180;
                                        if(parents[mutb].th2 >  180*pi/180) parents[mutb].th2 =
180*pi/180;
                                        else if(parents[mutb].th2 < -180*pi/180) parents[mutb].th2 = -
180*pi/180;
                                        if(parents[mutb].th3 >  180*pi/180) parents[mutb].th3 =
180*pi/180;
                                        else if(parents[mutb].th3 < -180*pi/180) parents[mutb].th3 = -
180*pi/180;
                                        if(parents[mutb].th4 >  180*pi/180) parents[mutb].th4 =
180*pi/180;
                                        else if(parents[mutb].th4 < -180*pi/180) parents[mutb].th4 = -
180*pi/180;
                                        if(parents[mutb].th5 >  180*pi/180) parents[mutb].th5 =
180*pi/180;
                                        else if(parents[mutb].th5 < -180*pi/180) parents[mutb].th5 = -
180*pi/180;
                                        if(parents[mutb].th6 >  180*pi/180) parents[mutb].th6 =
180*pi/180;
                                        else if(parents[mutb].th6 < -180*pi/180) parents[mutb].th6 = -
180*pi/180;
                                }

                                /*Armazenando qual foi o best do último melhor individuo*/
                                bestant = parents[best].fitness;

                                /*Atualizando o Cálculo do Tempo de Execução*/
                                ftime(&fim);

                                /*Atualizando o Cálculo do Numero de Gerações*/
                                geracoes_cont++;

                                /*Parando a simulação quando a precisão for atingida ou passar mais de 1
segundo*/
                                if(parents[best].erro < acerto || (((double)
fim.time+((double)fim.millitm*0.001))-((double)ini.time+((double)ini.millitm*0.001))) >= 1.0)
                                        break;
                        }

                        /*Armazenando Respostas*/
                        resposta[tries].x = parents[best].x;
                        resposta[tries].y = parents[best].y;
                        resposta[tries].z = parents[best].z;
                        resposta[tries].th1 = parents[best].th1;
                        resposta[tries].th2 = parents[best].th2;
                        resposta[tries].th3 = parents[best].th3;
                        resposta[tries].th4 = parents[best].th4;
                        resposta[tries].th5 = parents[best].th5;
                        resposta[tries].th6 = parents[best].th6;
                        resposta[tries].r11 = parents[best].r11;
                        resposta[tries].r12 = parents[best].r12;
                        resposta[tries].r13 = parents[best].r13;
                        resposta[tries].r21 = parents[best].r21;
                        resposta[tries].r22 = parents[best].r22;
                        resposta[tries].r23 = parents[best].r23;
                        resposta[tries].r31 = parents[best].r31;
                        resposta[tries].r32 = parents[best].r32;
                        resposta[tries].r33 = parents[best].r33;
                        resposta[tries].fitness = parents[best].fitness;
                        resposta[tries].erropos = parents[best].erropos;
                        resposta[tries].erroant = parents[best].erroant;
                        resposta[tries].erroori = parents[best].erroori;
```

```
                        resposta[tries].tempo = (((double) fim.time+((double)fim.millitm*0.001))-
((double)ini.time+((double)ini.millitm*0.001)));
                        resposta[tries].geracoes = geracoes_cont;
                }

                /*Escolhendo a melhor resposta*/
                best_resposta = 0;
                resposta_ant = 1000000;
                for(int tries=0;tries<samples;tries++){
                        if(resposta[best_resposta].fitness > resposta[tries].fitness)
                                best_resposta = tries;
                }

                /*Impressão dos resultados*/
                printf("\n\tTeste %d\n",
                        teste);

                printf("\nResultados dos cem:\n\n");
                if(mode == 0){
                        printf("\n[Tempo em s], [Iteracoes], [Erro Posicao]\n");
                        media_tempo = 0; desvpadr_tempo = 0;
                        media_interacoes = 0; desvpadr_interacoes = 0;
                        media_erropos = 0; desvpadr_erropos = 0;

                        for(int tries=0;tries<samples;tries++){
                                printf("%.3lf, %d, %.2lf\n",
                                resposta[tries].tempo, resposta[tries].geracoes, resposta[tries].erropos);
                                media_tempo = media_tempo + resposta[tries].tempo;
                                media_interacoes = media_interacoes + resposta[tries].geracoes;
                                media_erropos = media_erropos + resposta[tries].erropos;
                        }
                        media_tempo = media_tempo/samples;
                        media_interacoes = media_interacoes/samples;
                        media_erropos = media_erropos/samples;

                        for(int tries=0;tries<samples;tries++){
                                desvpadr_tempo = desvpadr_tempo + pow((resposta[tries].tempo -
media_tempo),2);
                                desvpadr_interacoes = desvpadr_interacoes + pow((resposta[tries].geracoes -
media_interacoes),2);
                                desvpadr_erropos = desvpadr_erropos + pow((resposta[tries].erropos -
media_erropos),2);
                        }
                        desvpadr_tempo = sqrt(desvpadr_tempo/(samples-1));
                        desvpadr_interacoes = sqrt(desvpadr_interacoes/(samples-1));
                        desvpadr_erropos = sqrt(desvpadr_erropos/(samples-1));

                        printf("\nMedias:\n");
                        printf("%.3lf, %.0lf, %.2lf\n",
                        media_tempo, media_interacoes, media_erropos);
                        printf("\nDesvio Padrao:\n");
                        printf("%.3lf, %.0lf, %.2lf\n",
                        desvpadr_tempo, desvpadr_interacoes, desvpadr_erropos);
                }
                if(mode == 1){
                        printf("\n[Tempo em s], [Iteracoes], [Erro Posicao], [Erro Angulo Inicial]\n");
                        media_tempo = 0; desvpadr_tempo = 0;
                        media_interacoes = 0; desvpadr_interacoes = 0;
                        media_erropos = 0; desvpadr_erropos = 0;
                        media_erroant = 0; desvpadr_erroant = 0;

                        for(int tries=0;tries<samples;tries++){
                                printf("%.3lf, %d, %.2lf, %.2lf\n",
                                resposta[tries].tempo, resposta[tries].geracoes, resposta[tries].erropos,
resposta[tries].erroant);
                                media_tempo = media_tempo + resposta[tries].tempo;
                                media_interacoes = media_interacoes + resposta[tries].geracoes;
```

```
                                media_erropos = media_erropos + resposta[tries].erropos;
                                media_erroant = media_erroant + resposta[tries].erroant;
                        }
                        media_tempo = media_tempo/samples;
                        media_interacoes = media_interacoes/samples;
                        media_erropos = media_erropos/samples;
                        media_erroant = media_erroant/samples;

                        for(int tries=0;tries<samples;tries++){
                                desvpadr_tempo = desvpadr_tempo + pow((resposta[tries].tempo -
media_tempo),2);
                                desvpadr_interacoes = desvpadr_interacoes + pow((resposta[tries].geracoes -
media_interacoes),2);
                                desvpadr_erropos = desvpadr_erropos + pow((resposta[tries].erropos -
media_erropos),2);
                                desvpadr_erroant = desvpadr_erroant + pow((resposta[tries].erroant -
media_erroant),2);
                        }
                        desvpadr_tempo = sqrt(desvpadr_tempo/(samples-1));
                        desvpadr_interacoes = sqrt(desvpadr_interacoes/(samples-1));
                        desvpadr_erropos = sqrt(desvpadr_erropos/(samples-1));
                        desvpadr_erroant = sqrt(desvpadr_erroant/(samples-1));

                        printf("\nMedias:\n");
                        printf("%.3lf, %.0lf, %.2lf, %.2lf\n",
                        media_tempo, media_interacoes, media_erropos, media_erroant);
                        printf("\nDesvio Padrao:\n");
                        printf("%.3lf, %.0lf, %.2lf, %.2lf\n",
                        desvpadr_tempo, desvpadr_interacoes, desvpadr_erropos, desvpadr_erroant);
                }
                if(mode == 2){
                        printf("\n[Tempo em s], [Iteracoes], [Erro Posicao], [Erro Orientacao]\n");
                        media_tempo = 0; desvpadr_tempo = 0;
                        media_interacoes = 0; desvpadr_interacoes = 0;
                        media_erropos = 0; desvpadr_erropos = 0;
                        media_erroori = 0; desvpadr_erroori = 0;

                        for(int tries=0;tries<samples;tries++){
                                printf("%.3lf, %d, %.2lf, %.2lf\n",
                                resposta[tries].tempo, resposta[tries].geracoes, resposta[tries].erropos,
resposta[tries].erroori);
                                media_tempo = media_tempo + resposta[tries].tempo;
                                media_interacoes = media_interacoes + resposta[tries].geracoes;
                                media_erropos = media_erropos + resposta[tries].erropos;
                                media_erroori = media_erroori + resposta[tries].erroori;
                        }
                        media_tempo = media_tempo/samples;
                        media_interacoes = media_interacoes/samples;
                        media_erropos = media_erropos/samples;
                        media_erroori = media_erroori/samples;

                        for(int tries=0;tries<samples;tries++){
                                desvpadr_tempo = desvpadr_tempo + pow((resposta[tries].tempo -
media_tempo),2);
                                desvpadr_interacoes = desvpadr_interacoes + pow((resposta[tries].geracoes -
media_interacoes),2);
                                desvpadr_erropos = desvpadr_erropos + pow((resposta[tries].erropos -
media_erropos),2);
                                desvpadr_erroori = desvpadr_erroori + pow((resposta[tries].erroori -
media_erroori),2);
                        }
                        desvpadr_tempo = sqrt(desvpadr_tempo/(samples-1));
                        desvpadr_interacoes = sqrt(desvpadr_interacoes/(samples-1));
                        desvpadr_erropos = sqrt(desvpadr_erropos/(samples-1));
                        desvpadr_erroori = sqrt(desvpadr_erroori/(samples-1));

                        printf("\nMedias:\n");
```

```
                        printf("%.3lf, %.0lf, %.2lf, %.2lf\n",
                        media_tempo, media_interacoes, media_erropos, media_erroori);
                        printf("\nDesvio Padrao:\n");
                        printf("%.3lf, %.0lf, %.2lf, %.2lf\n",
                        desvpadr_tempo, desvpadr_interacoes, desvpadr_erropos, desvpadr_erroori);
                }

                printf("\nResultados do melhor:\n\n");
                printf("\nPosicao (em cm) Desejada:\nX = %.2lf, Y = %.2lf, Z = %.2lf\n",
                        x, y, z);
                printf("\nPosicao (em cm) Encontrada:\nX = %.2lf, Y = %.2lf, Z = %.2lf\n",
                        resposta[best_resposta].x, resposta[best_resposta].y, resposta[best_resposta].z);

                if(mode == 1){
                        printf("\nAngulos (em graus) Iniciais:\nth1: %.2lf, th2: %.2lf, th3: %.2lf, th4: %.2lf,
th5: %.2lf, th6: %.2lf\n",
                                thini1*180/pi, thini2*180/pi, thini3*180/pi, thini4*180/pi, thini5*180/pi,
thini6*180/pi);
                }
                printf("\nAngulos (em graus) Encontrados:\nth1: %.2lf, th2: %.2lf, th3: %.2lf, th4: %.2lf, th5:
%.2lf, th6: %.2lf\n",
                        resposta[best_resposta].th1*180/pi, resposta[best_resposta].th2*180/pi,
resposta[best_resposta].th3*180/pi,
                        resposta[best_resposta].th4*180/pi, resposta[best_resposta].th5*180/pi,
resposta[best_resposta].th6*180/pi);

                if(mode == 2){
                        printf("\nMatriz de Rotacao Row, Pitch Yaw:\n");
                        printf("[%.2lf, %.2lf, %.2lf]\n[%.2lf, %.2lf, %.2lf]\n[%.2lf, %.2lf, %.2lf]\n",
                                resposta[best_resposta].r11, resposta[best_resposta].r12,
resposta[best_resposta].r13,
                                resposta[best_resposta].r21, resposta[best_resposta].r22,
resposta[best_resposta].r23,
                                resposta[best_resposta].r31, resposta[best_resposta].r32,
resposta[best_resposta].r33);
                }

                if(mode == 0){
                        printf("\nErro da Posicao: %.2lf\n",
                                resposta[best_resposta].erropos);
                }
                if(mode == 1){
                        printf("\nErro da Posicao: %.2lf. Erro dos Angulos Iniciais: %.2lf\n",
                                resposta[best_resposta].erropos, resposta[best_resposta].erroant);
                }
                if(mode == 2){
                        printf("\nErro da Posicao: %.2lf. Erro da Orientacao: %.2lf\n",
                                resposta[best_resposta].erropos, resposta[best_resposta].erroori);
                }
                printf("\nO calculo foi realizado em apoximadamente %.3fs, e em %d iteracoes\n\n",
                        resposta[best_resposta].tempo, resposta[best_resposta].geracoes);
        }
        //getchar();
        return 0;
}
```