UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA MECÂNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA MECÂNICA

FERNANDO CÉSAR DE CERQUEIRA JÚDICE TAVARES

UTILIZAÇÃO DE UM SISTEMA MULTI MANIPULADOR SCARA PARA AUMENTAR O VOLUME DE TRABALHO E REDUZIR O TEMPO DE OPERAÇÃO NO PROCESSO DE FABRICAÇÃO POR DEPÓSITO DE FIO FUNDIDO

Recife
2019

FERNANDO CÉSAR DE CERQUEIRA JÚDICE TAVARES

UTILIZAÇÃO DE UM SISTEMA MULTI MANIPULADOR SCARA PARA AUMENTAR O VOLUME DE TRABALHO E REDUZIR O TEMPO DE OPERAÇÃO NO PROCESSO DE FABRICAÇÃO POR DEPÓSITO DE FIO FUNDIDO

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Mecânica da Universidade Federal de Pernambuco como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Pedro Manuel González del Foyo.

Recife

2019

Catalogação na fonte
Bibliotecário Josias Machado, CRB-4 / 1690

T231u Tavares, Fernando César de Cerqueira Júdice.
Utilização de um sistema multimanipulador SCARA para aumentar o volume de trabalho e reduzir o tempo de operação no processo de fabricação por depósito de fio fundido / Fernando César de Cerqueira Júdice Tavares. – Recife, 2019.
107 folhas, il., figs., tabs.

Orientador: Prof. Dr. Pedro Manuel González del Foyo.

TCC (Graduação) – Universidade Federal de Pernambuco. CTG. Departamento de Engenharia Mecânica, 2019.
Inclui Referências e apêndices.

1. Engenharia Mecânica. 2. Impressão 3d. 3. Fabricação aditiva. 4. Manipulador SCARA. 5. Sistemas multi manipulador. 6. Coordenação de movimento. I. del Foyo, Pedro Manuel González (orientador). II. Título.

UFPE

621 CDD (22. ed.) BCTG/2019-330

FERNANDO CÉSAR DE CERQUEIRA JÚDICE TAVARES

UTILIZAÇÃO DE UM SISTEMA MULTI MANIPULADOR SCARA PARA AUMENTAR O VOLUME DE TRABALHO E REDUZIR O TEMPO DE OPERAÇÃO NO PROCESSO DE FABRICAÇÃO POR DEPÓSITO DE FIO FUNDIDO

Trabalho de Conclusão de Curso apresentado ao departamento de engenharia mecânica da Universidade federal de Pernambuco como parte dos requisitos necessários para a obtenção do Grau de Bacharel em engenharia mecânica.

.

Aprovado em: 05/07/19.

BANCA EXAMINADORA

_____

Prof. Dr. Pedro Manuel González del Foyo (Orientador)
Universidade Federal de Pernambuco

_____

Prof. Dr. João Paulo Cerquinho Cajueiro (Examinador Interno)
Universidade Federal de Pernambuco

_____

Prof. Dr. Guaraci Guimarães Bastos Junior (Examinador Interno)
Universidade Federal de Pernambuco

RESUMO

O processo de fabricação por depósito de fio fundido sempre encontrou limitações no tamanho das peças a serem fabricadas. Tais limitações se apresentam tanto no espaço disponível para utilização quanto na viabilidade de tempo. A utilização de manipuladores alternativos ao *Gantry* se mostrou viável para solucionar o problema do espaço. Combinado a isto, a capacidade desses manipuladores atuarem de maneira coordenada permitiu também propor uma solução para o problema do tempo. O presente trabalho teve como objetivo, primeiramente, avaliar o ganho em volume de trabalho ao trocar do manipulador *Gantry* para o SCARA. Em segunda instância, foi proposto um algorítimo baseado em uma adaptação do método de *Grid* capaz de coordenar dois manipuladores trabalhando em uma mesma peça. Inicialmente, foi apresentado um breve contexto ressaltando as dificuldades e virtudes dos diferentes manipuladores com destaque àqueles comumente utilizados na fabricação aditiva. Em seguida, foram explicitados os fundamentos nos quais o algorítimo se baseia e foram propostos dois casos de estudo. Por fim, concluiu-se com a ponderação dos resultados obtidos tanto a respeito do volume de trabalho quanto à divisão de tarefas entre os manipuladores. Concluiu-se que o algorítimo é promissor, com potencial de ser expandido a peças mais complexas. Todas as simulações numéricas foram realizadas no programa *Matlab*.

Palavras-chave**:** Impressão 3D. Fabricação aditiva. Manipulador SCARA. Sistemas multi manipulador. Coordenação de movimento.

ABSTRACT

3D printing through melted plastic filament wire has always faced the barriers of limited workspace and long task times. Switching from the traditional Gantry manipulator to a more dexterous is an intuitive approach to solve the limited space problem. In addition, it is often possible to combine multiple manipulators into a single system through coordinated motion planning thus potentially solving the time constraints. The present work has as its main goal to propose and simulate a SCARA setting as an alternative to Gantry manipulators. Following that premise, it will also be proposed an algorithm to coordinate a multi manipulator system consisting of two SCARA robots. Firstly, a brief background regarding manipulators applied to additive manufacturing will be headed to the reader. Secondly, the guidelines for the algorithm will be given, and two cases will be proposed in order to validate it. Finally, the results are shown and it will evaluate both the workspace gain for the new setup and the efficiency of the coordination algorithm through the work cases. The software *matlab* will be used as aid during all simulations.

Keyword: 3D Printing. Additive manufacturing. SCARA Manipulator. Multi Manipulator Systems. Coordinated motion planning.

SUMÁRIO

1 INTRODUÇÃO 9

1.1 JUSTIFICATIVA 11

1.2 OBJETIVOS 13

1.2.1 Objetivo geral 13

1.2.2 Objetivos específicos 13

2 FUNDAMENTAÇÃO TEÓRICA 14

2.1 TIPOS DE MANIPULADORES 14

2.1.1 Manipulador cartesiano 14

2.1.2 Manipulador cilíndrico 14

2.1.3 Manipulador planar 15

2.1.4 Manipulador articulado 15

2.1.5 Manipulador SCARA 16

2.2 ANÁLISE CINEMÁTICA 16

2.2.1 Movimento de corpo rígido 18

2.2.2 Convenção de Denavit-Hartenberg 20

2.2.3 Cinemática direta e indireta 23

2.2.3.1 Cinemática direta 24

2.2.3.2 Cinemática direta do manipulador SCARA 24

2.2.3.3 Cinemática inversa 26

2.2.3.4 Cinemática inversa para o manipulador SCARA 27

2.3 VOLUME DE TRABALHO DO SCARA IDEAL 28

2.3.1 Singularidade 29

2.4 GERAÇÃO DE TRAJETÓRIA 34

3 METODOLOGIA 36

3.1 DIMENSIONAMENTO GEOMÉTRICO DO MANIPULADOR 36

3.1.1 Posicionamento multi manipulador 42

3.2 VERIFICAÇÃO DE COLISÃO 42
3.3 COORDENAÇÃO DOS MANIPULADORES 46
3.3.1 Definição das tarefas 49
3.3.2 Escolha da célula 50
3.3.3 Divisão da trajetória 53
3.3.4 Malha inicial 54
3.3.5 Movimento entre tarefas 60
3.3.6 Divisão do contorno 60
3.4 CASOS DE ESTUDO 62
4 ANÁLISE DOS RESULTADOS 65
4.1 VERIFICAÇÃO NUMÉRICA DA ÁREA DE TRABALHO 65
4.2 CASOS DE ESTUDO 66
4.2.1 Caso 1 66
4.2.2 Caso 2 70
4.2.3 Análise do tempo 72
5 CONCLUSÃO 75
REFERÊNCIAS 76
APÊNDICE A – CÓDIGO PRINCIPAL 78
APÊNDICE B - CÓDIGO DADOS DE ENTRADA 93
APÊNDICE C - FUNÇÃO INFILL 94
APÊNDICE D - FUNÇÃO CINEMÁTICA INVERSA 96
APÊNDICE E - FUNÇÃO POSIÇÃO DO MANIPULADOR 97
APÊNDICE F - FUNÇÃO ORDENAR PONTOS DO INFILL 98
APÊNDICE G - FUNÇÃO DIVIDIR CAMINHO 99
APÊNDICE H - FUNÇÃO TEMPO DE EXECUÇÃO 101
APÊNDICE I - FUNÇÃO GERAR GEOMETRIA DO BRAÇO 102
APÊNDICE J - FUNÇÃO AVALIAR COLISÃO 103

APÊNDICE K - FUNÇÃO COLISÃO PONTO-MANIPULADOR 104

APÊNDICE L - FUNÇÃO COLISÃO RETA COM MANIPULADOR 105

APÊNDICE M - FUNÇÃO PLOT BRAÇO DO MANIPULADOR 107

1 INTRODUÇÃO

Atualmente o termo *impressão 3D* foi vulgarizado na mídia e é comumente utilizado para designar todo e qualquer método de fabricação aditiva. Todavia, Horvath (2014) afirma que a impressão 3D constitui apenas uma parte desse universo que é a fabricação aditiva. Tal cenário engloba muitas técnicas, mas todas se juntam em um núcleo comum: a construção de uma forma tridimensional a partir da soma de finas camada de pequena espessura. Neste trabalho, o termo impressão 3D será utilizado como sinônimo de manufatura aditiva.

As primeiras patentes de fabricação aditiva datam do início da década de 80. Foi nesse período em que o projeto assistido por computador (CAD) se estabeleceu, bem como o uso dos comandos numéricos programáveis. O terceiro componente crítico foi o avanço nas pesquisas de novos materiais, tais quais os polímeros diversos e resinas fotossensíveis. O conjunto desses fatores definiu a massa crítica necessária para o desenvolvimento dessa técnica de fabricação.

Dentre os métodos, encontra-se a categoria de fabricação por extrusão de material. Esses processos utilizam um bico extrusor para expelir um fino filamento de um material determinado, e deposita esse material de tal maneira a dar a forma desejada (KAMRAN, 2016). Dentre eles, um processo merece destaque, o método de fabricação por depósito de fio fundido, desenvolvido em 1989 por Scott Crump.

O método se beneficia da baixa temperatura de fusão dos polímeros. Um fio de material polimérico é empurrado contra um bico aquecido. O material se liquefaz pouco antes de ser expelido e se solidifica pouco depois de ser ejetado. Conforme o material é liberado, um manipulador posiciona o bico. O fio depositado ao longo da trajetória do manipulador irá gerar a peça.

Historicamente, os manipuladores do tipo cartesiano foram os mais utilizados. Tal escolha é bem justificada. Sciavicco (2006) exalta que os manipuladores cartesianos alcançam precisões maiores mais facilmente uma vez que um movimento de uma junta implica no mesmo movimento da ferramenta. O controle desse tipo de sistema é comumente realizado em malha aberta, o que reduz os custos tanto em processamento quanto em receptores eletrônicos.

Dentre os tipos de robôs cartesianos, o tipo *gantry* é posto em evidência. Tal configuração pode ser vista na impressora prusa i3, responsável por popularizar o mercado doméstico de fabricação aditiva, e na *the buccaneer*, impressora que levantou 1,5 milhões de dólares em 2013 (HORVATH, 2014). O maior benefício, como cita Sciavicco (2006), é a alta

rigidez dos manipuladores gantry, podendo-se interpretar rigidez nesse contexto como sinônimo de robustez. A precisão do sistema aumenta e o sistema de controle se torna ainda mais fácil.

A simplicidade de projeto popularizou os manipuladores cartesianos na impressão 3D. Porém, sua utilização tem seus ônus. Dentre eles está a ineficiência com relação ao espaço utilizado. Craig (1989) evidencia que juntas prismáticas ocupam mais espaço com relação ao ambiente de trabalho que juntas de revolução. Para manipuladores tipo gantry, a geometria é ainda mais restritiva: A estrutura do manipulador é sempre maior do que seu volume de trabalho.

No que tange as técnicas de fabricação por depósito de filamento fundido, essa ineficiência resulta em consequências significantes. A primeira é a dificuldade em dimensionar a máquina para volumes de trabalho maiores. Um aumento no espaço de atuação necessita que o manipulador aumente de, no mínimo, o mesmo tanto. Isso implica em estruturas maiores, perda de rigidez e necessidade de atuadores de maior potência.

As restrições técnicas não são o único obstáculo para o projeto de máquinas maiores. O manipulador gantry, por envolver o volume de trabalho, restringe aquele espaço permanentemente. Ou seja, o mesmo volume deve ser deslocado mesmo se utilizado apenas uma fração do espaço de atuação total da impressora. Essa ineficiência é amplificada nos momentos em que o aparelho não estiver em uso, sendo bastante taxativo, em especial, para o mercado doméstico.

Colocando o mercado doméstico em evidência, surge outra dificuldade. É comum nesse contexto realizar modificação e reposição de peças utilizando a própria impressora. Dessa forma, há um grande interesse em ter o maior número de componentes possíveis capazes de serem fabricados pelo próprio sistema (OGULMUS, 2016). Caso a estrutura seja intrinsecamente maior que o espaço de trabalho disponível, fica inviável sua produção.

Entretanto, uma das presentes limitações do método de fabricação por depósito de fio fundido é o tempo de fabricação. Horvath(2014) comenta que o tempo de impressão pode ir de horas até dias. Aumentar o volume de trabalho sem preocupação com o tempo apenas amplificaria esse problema.

A solução mais simplista seria aumentar a potência dos atuadores. Todavia, essa proposta esbarra em algumas barreiras. A maior delas está relacionada ao processo de extrusão do fio polimérico. Berlini (2004) associa a velocidade de extrusão com a pressão do filamento derretido no bico extrusor. Quanto mais rápido for a liberação do fio, maior deve ser a pressão e consequentemente requer um atuador de maior potência.

Dessa forma, não é interessante tentar solucionar o problema do tempo apenas através do aumento da velocidade individual do manipulador. Seria conveniente que o manipulador proposto fosse mecanicamente o mais similar possível com a tecnologia atual tanto para aproveitar as peças já popularizadas no mercado quanto para utilizar todo o ecossistema de programas e ferramentas desenvolvidos para os métodos atuais.

1.1 JUSTIFICATIVA

Quando a otimização do espaço de trabalho torna-se um critério, a utilização de juntas de revolução é vantajosa. Como destaca Sciavicco (2006), pares cinemáticos de revolução são preferidos quando a compactabilidade é um critério importante. Dessa forma, ao se reduzir o número de juntas prismáticas, é possível propor uma nova solução capaz de atender aos critérios de espaço. Assim, uma mudança de manipulador é uma solução viável para o problema.

A utilização de outras configurações não é novidade. Outros tipos de manipuladores já foram propostos na tentativa de sanar outros problemas das impressoras do tipo gantry. Em destaque está a configuração em delta, a qual permite elevar a velocidade de impressão. Esse tipo de impressora ganhou sensível destaque e é um dos poucos além da gantry a ser produzido em escala.

Ao priorizar o espaço de trabalho, a escolha do manipulador tem influência em um conjunto de aspectos relevantes. O primeiro ponto é a forma do volume de trabalho. A geometria é capaz de mudar bruscamente entre uma configuração e outra, podendo assumir desde um formato esférico até uma forma cúbica. O segundo elemento a ser destacado é o conjunto de orientações que a ferramenta pode assumir em um dado ponto do espaço. Ambas as características devem ser otimizadas para cada tarefa e vão ditar a seleção inicial do tipo de manipulador.

Uma vez que os métodos de fabricação aditiva trabalham com a deposição de camadas na direção vertical, há um interesse em manter as sessões do volume de trabalho constantes nessa orientação. Ter um volume de trabalho regular facilita o posicionamento dos objetos a serem fabricados. É preferível, então, volumes prismáticos em detrimento dos esféricos.

A necessidade de variar a orientação do manipulador está intrinsecamente relacionada com o método de fabricação aditiva utilizado. O processo de fabricação por depósito de fio fundido, em particular, não tem interesse em uma elevada destreza da ferramenta. O contrário

é verdade: o método se beneficia em manter a ferramenta sempre com a mesma orientação vertical com relação à superfície de trabalho.

A configuração que satisfaz essas condições, além do manipulador cartesiano, é o modelo SCARA (*selective compliance assembly robot arm*), o qual é composto por um eixo prismático na direção vertical e um manipulador 2D formado por duas juntas de revolução. Sua utilização não é novidade. Okabe (2016) propôs um robô SCARA de quatro braços para impressão 3D. Ogulmus (2016) estudou o sistema de controle desse tipo de manipulador para o mesmo fim. Todavia, a relevância desse sistema para a fabricação aditiva ainda é limitada, com maior expressão no mercado doméstico e em projetos de hobistas.

A primeira característica a ser exaltada é que, ao utilizar pares de revolução, o alcance máximo do manipulador em um determinado sentido é definido por uma associação de múltiplos elos, ao passo que juntas prismáticas dependem de uma guia tão grande quanto a distância. Isso implica que a distância varrida é consideravelmente maior que o tamanho das peças individuais que compõem a cadeia. A redução do tamanho dos elementos da estrutura viabiliza a confecção utilizando o próprio método de fabricação aditiva.

A mudança permite, também, realizar a separação entre a estrutura de confinamento e o braço do manipulador. Isso significa que o manipulador pode ser projetado para atender volumes de trabalhos diversos, e que variem ao longo do tempo de vida da máquina. Tal comportamento pode ser obtido alterando apenas os limites de movimento das articulações, sem mudanças nos componentes físicos da máquina. O equipamento resultado.

As restrições de dimensionamento no projeto de máquinas de maior volume ainda existem para manipuladores do tipo SCARA. Porém, dispõe-se também de ferramentas para contornar o problema. Composições em paralelo permitem elevar a força as juntas e melhorar a rigidez alterando apenas a conexão entre as peças (SPONG, 2006). Em adição, é possível o acionamento da junta de maneira indireta, o que permite melhor distribuição do peso ao longo do braço.

A mudança de manipulador apenas, apesar de favorecer o volume de impressão, não soluciona o problema do tempo. Tavares (2016) coloca a redução do tempo como uma das vantagens de sistemas multi manipuladores. De fato, em um cenário ideal, a utilização de dois manipuladores reduziria o tempo de impressão em aproximadamente 50%.

Utilizar múltiplos manipuladores permite reduzir o tempo de total de impressão sem a necessidade de aumentar a velocidade individual de cada máquina. Dessa forma, é possível conseguir um ganho considerável de eficiência e ao mesmo tempo contornar as limitações mecânicas da extrusão do filamento.

1.2 OBJETIVOS

O presente trabalho é centrado em um objetivo geral, o qual pode ser discriminado em um conjunto de objetivos específicos.

1.2.1 Objetivo geral

- Propor um sistema multi manipulador SCARA para aumentar o volume de trabalho e reduzir o tempo de impressão no processo de manufatura por fil fundido.

1.2.2 Objetivos específicos

- Projetar um manipulador tipo SCARA de maneira a otimizar o volume de trabalho.
- Desenvolver um algorítimo de coordenação de tarefas para sistemas multi manipuladores
- Utilizar os manipuladores SCARA obtidos em consonância com o algorítmo para gerar trajetórias de fabricação das peças.
- Estimar o ganho de tempo esperado utilizando múltiplos manipuladores.

2 FUNDAMENTAÇÃO TEÓRICA

O escopo de conhecimento necessário para o estudo da manufatura aditiva é consideravelmente extenso. É preciso entender os tipos de manipuladores e os modelos matemáticos aplicáveis, bem como as possíveis abordagens para o comando e controle da máquina.

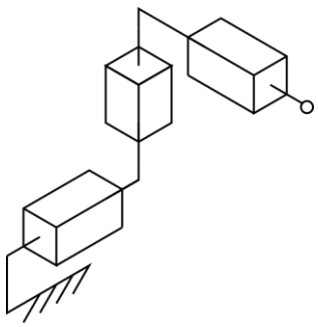
2.1 TIPOS DE MANIPULADORES

São vários os tipos de manipuladores encontrados no mercado. Algumas configurações, porém, merecem destaque pelo seu uso (ROSÁRIO, 2005). Dentre eles, serão evidênciados manipuladores seriais. Configurações em paralelo, em primeira instância, não serão de interesse pois seu volume de trabalho é, naturalmente, mais restrito.

2.1.1 Manipulador cartesiano

O manipulador cartesiano é composto por três juntas prismáticas ortogonais entre si. É conhecido pela robustez e alta precisão. Todavia, sua destreza é baixa. Na maioria dos casos, está restrito a três graus de liberdade, mantendo a orientação da ferramenta fixa durante todo o funcionamento. Seu volume de trabalho tem formato retangular.

Figura 1 - Manipulador cartesiano



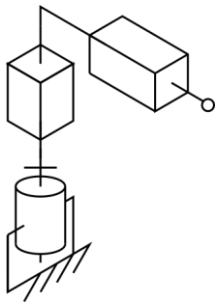
Fonte: O Autor (2019).

2.1.2 Manipulador cilíndrico

O manipulador cilíndrico possui uma rótula em sua base, permitindo rotação no plano paralelo a superfície. As outras duas juntas são prismáticas, na vertical e horizontal. A

principal diferença para o manipulador cartesiano é o volume de trabalho, que é um cilindro vazado.

Figura 2 - Manipulador cilíndrico

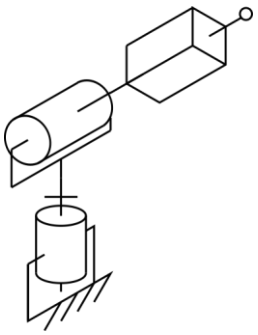


Fonte: O Autor (2019).

2.1.3 Manipulador planar

O manipulador planar é composto de uma junta prismática na extremidade da ferramenta e de duas juntas de revolução. Seu volume de trabalho é esférico e ele é capaz de assumir diversas orientações para a ferramenta. Todavia, é menos robusto que os modelos anteriores e requer um controle sensivelmente mais fino.

Figura 3 - Manipulador planar

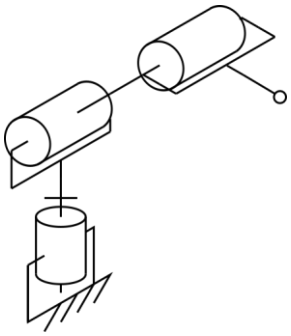


Fonte: O Autor (2019).

2.1.4 Manipulador articulado

Dentre as configurações, é a que possui maior destreza. É capaz de assumir um elevado número de configurações dentro de seu volume de trabalho esférico. Sua rigidez, porém, é significativamente baixa. Sistemas de controle complexos são fundamentais para assegurar seu bom funcionamento.

Figura 4 - Manipulador articulado

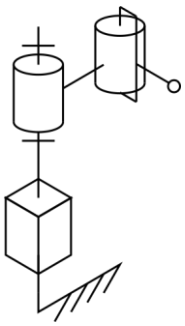


Fonte: O Autor (2019).

2.1.5 Manipulador SCARA

O modelo SCARA (*selective compliance assembly robot arm*) é semelhante ao planar, diferindo apenas na ordem com que as juntas se revelam. A junta prismática se encontra na direção vertical, e as outras duas juntas de revolução são responsáveis pelo movimento paralelo a superfície. São robustos, mas têm baixa destreza. Seu volume de trabalho tem formato prismático.

Figura 5 - Manipulador SCARA



Fonte: O Autor (2019).

2.2 ANÁLISE CINEMÁTICA

A compreensão de um manipulador está diretamente relacionada com o entendimento de cadeias cinemáticas. Sciavicco (2006), Spong(2005) e Siciliano (2016), evidenciam que a estrutura fundamental de um manipulador é uma associação de juntas e elos em cadeia. Atrelado a esses conceitos, está a ideia de mobilidade e graus de liberdade. Todas essas grandezas e componentes precisam ser bem definidos para viabilizar a análise.

Norton (2010) afirma que a mobilidade de um sistema mecânico é caracterizada de acordo com o número de graus de liberdade que possui. Os graus de liberdade (GDL) transcrevem a quantidade de parâmetros independentes necessários para caracterizar uma única posição no espaço. Quanto maior a mobilidade do sistema, mais posições e orientações o mesmo pode assumir.

Seis graus de liberdade são suficientes para permitir ao manipulador assumir qualquer configuração no espaço. Nem sempre, todavia, a mobilidade completa é necessária para os processos de fabricação aditiva. Alguns métodos, como a estereolitografia, se beneficiam de manipular a orientação da ferramenta. A maioria, incluindo o método de depósito de fio fundido, necessita apenas do controle da posição mantendo a orientação do atuador constante durante todo o processo.

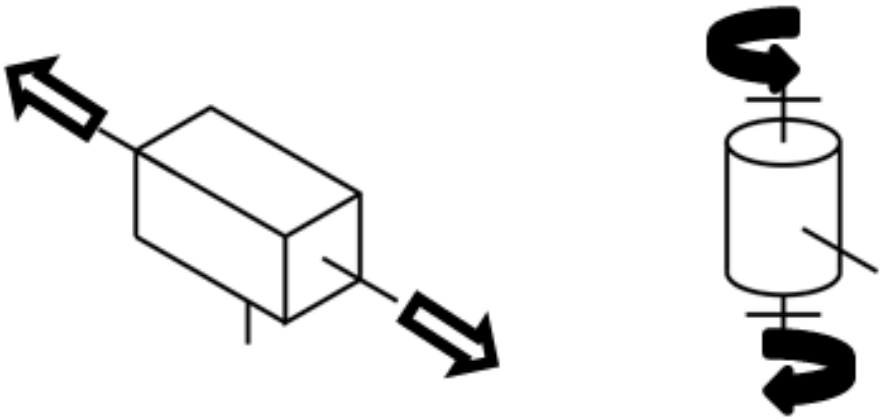
Uma cadeia cinemática é composta pela associação de elos e juntas. Um elo é um corpo rígido o qual possui ao menos dois nós para se conectar a outros corpos. Uma junta ou par cinemático é, por sua vez, uma conexão entre dois ou mais elos (NORTON, 2010). Lynch (2017) afirma que uma junta pode ser interpretada como uma maneira de adicionar mobilidade ao sistema. Consequentemente, a maneira como elas conectam os elos ao longo da cadeia irá determinar a mobilidade do manipulador.

Diversos tipos de juntas podem ser encontrados na engenharia. Norton (2010) coloca o número de graus de liberdade como uma maneira conveniente de classificá-las. O número de GDL de uma junta indica a mobilidade do elo associado a ela. Assim, um par cinemático com um GDL restringe o elo de cinco de seus seis graus de liberdade no espaço.

As juntas podem variar desde um até seis GDL. Durante a análise cinemática, entretanto, é conveniente simplicar os pares mais complexos transformando-os em uma associação de juntas com menos GDL. Segundo afirma Sciavicco (2006), todo par cinemático pode ser reduzido, sem perda de generalidade, a uma associação de juntas com um GDL e distância nula entre elas.

Dois pares cinemáticos, então, mantém relevância para o estudo: As juntas prismáticas e de revolução. A primeira permite movimento linear relativo entre os dois elos, enquanto a última permite rotação entre dois elos. O par helicoidal também é citado na literatura por possuir um grau de liberdade (Lynch, 2017). Seu uso, porém, é bastante limitado, sendo comumente substituído por uma associação de par prismático e de revolução.

Figura 6 - Junta prismática (esquerda) e de revolução (direita)



Fonte: O Autor (2019).

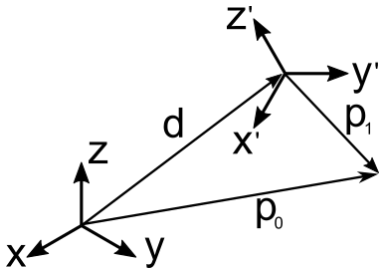
2.2.1 Movimento de corpo rígido

O interesse primário consiste em representar um corpo rígido no espaço com relação a um sistema de coordenadas qualquer. O corpo ocupa um volume, que é constituído por um conjunto de pontos. Assim, para conhecer totalmente o corpo, seria necessária informações sobre as posições de cada um dos ponto com relação à origem.

Craig (1987), todavia, aponta que a característica cerne de um corpo rígido é a de que não há movimento relativo entre os pontos dele. Dessa forma, uma vez que são conhecidas informações suficientes de apenas um dos pontos, é possível determinar todas as características dos demais. Ao associar, então, um sistema de coordenadas a um ponto arbitário fixo com relação ao corpo (sistema local), o problema de descreve-lo se resume a descrever o comportamento de tal sistema de coordenada com relação a um referencial (sistema de origem).

A transformação de um sistema de coordenada local com relação a um sistema de origem fixo pode ser descrita como a combinação de uma rotação pura com uma translação pura (SPONG, 2005). A rotação indica a movimentação angular entre os eixos principais dos sistemas, e é representada algebricamente como uma matriz de rotação 3x3. O deslocamento mensura o quanto a origem de um sistema se distancia do outro, e é representado algebricamente como um vetor 3x1.

Figura 7 - Relação entre dois sistemas de coordenadas



Fonte: O Autor (2019).

Cada ponto do corpo tem conhecida sua distância *p1* com o sistema local. A distância com relação ao sistema de origem pode ser obtida realizando uma operação vetorial.

$$p_0 = Rp_1 + d \tag{1}$$

No qual

$$R = Matriz\ de\ rotação$$
$$d = Vetor\ deslocamento$$

Spong (2005) descreve os dois movimentos em uma só matriz 4x4 da forma:

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix}$$

A matriz H é chamada **matriz de transformação homogênea** e caracteriza um **movimento homogêneo** no espaço. A última linha da matriz é uma linha auxiliar com a função de facilitar as operações matriciais ao torná-la uma matriz quadrada. Assim, pode-se obter a mesma relação da equação 1 através da operação da equação 2.

$$\begin{bmatrix} p_0 \\ 1 \end{bmatrix} = H \begin{bmatrix} p_1 \\ 1 \end{bmatrix} \tag{2}$$

Novamente, a última linha nos vetores p_0 e p_1 foram acrescidas pela consistência das operações.

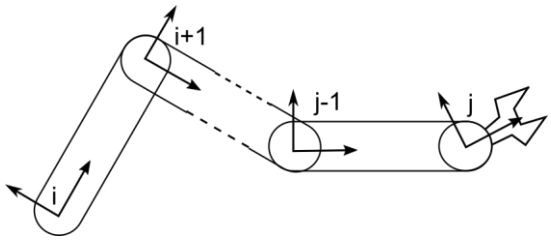
Ao representar elos ligados em série, a matriz homogênea se mostra deveras útil. Cada corpo da cadeia possui um sistema de coordenada local. Assim, a relação de um elo com outro pode ser representada por uma transformação homogênea. Nesse contexto, duas propriedades se destacam.

Primeiramente, cada vetor coluna possui uma interpretação segundo o espaço. Os três primeiros, associados a matriz de rotação, estão ligados a orientação do sistema local com relação à origem. A primeira coluna fornece a orientação segundo o eixo x da origem. A segunda apresenta a orientação segundo o eixo y, e a terceira, com relação ao eixo z (SPONG,

2005). Dessa forma, as três primeiras colunas indicam a orientação do corpo com relação ao sistema de origem, enquanto que a quarta coluna representa sua posição.

A segunda propriedade de interesse é a associação de múltiplas operações de transformação. É comum lidar com sistemas compostos por múltiplos corpos rígidos, cada qual com seu próprio sistema de coordenadas local. Uma vez que, em uma cadeia, a posição de um corpo depende dos anteriores, a relação entre corpos consecutivos é significativamente mais fácil de encontrar do que a relação de um corpo com o sistema global.

Figura 8 - Cadeia cinemática



Fonte: O Autor (2019).

$$H_j^i = H_{i-1}^i H_{i-2}^{i-1} \dots H_{j-1}^{j-2} H_j^{j-1} \tag{3}$$

Na qual

$$H_j^i = Transformação\ do\ sistema\ local\ i\ para\ o\ sistema\ de\ origem\ j.$$

Resta agora definir quantas informações são necessárias para bem representar uma determinada matriz de transformação homogênea. Lynch(2017) traz que com seis variáveis, qualquer matriz homogênea pode ser definida. De fato, como foi visto, uma transformação homogênea é caracterizada por uma rotação pura e uma translação pura. A rotação no espaço tridimensional pode ser sempre representada por três ângulos, enquanto que o deslocamento é representado por três coordenadas, totalizando seis variáveis.

2.2.2 Convenção de Denavit-Hartenberg

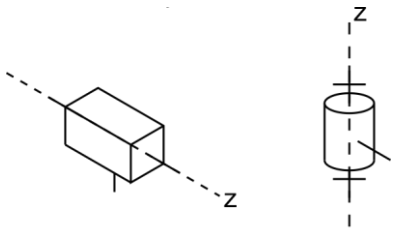
Em um caso geral, seis variáveis são necessárias para representar uma transformação homogênea. Todavia, Rosário (2005) afirma que a seleção inteligente dos sistemas de coordenadas pode reduzir esse número. A convenção de Denavit-Hartenberg para manipuladores permite reduzir o número de variáveis de seis para quatro. Tal convenção é

utilizada até os dias atuais em sistemas de corpos rígidos, como evidenciada no trabalho de Siciliano (2016) e Lynch (2017).

Deseja-se atribuir a cada elo um sistema de coordenadas que minimize o número de variáveis necessárias para descrevê-lo. Nesse contexto, a convenção de Denavit-Hartenberg (DH) abusa do fato que todas as juntas de um manipulador podem ser resumidas a uma associação de juntas prismáticas e/ou de revolução. A cada junta, será associado um sistema de coordenada. O sistema referencial será a junta da base do manipulador, recebendo o índice 0. Na outra extremidade da cadeia, estará o atuador.

Para cada junta, primeiramente, é definido o eixo z, que tem orientação dada pelo tipo dela. Juntas prismáticas tem o eixo z paralelo ao eixo de translação, enquanto que juntas de revolução tem o eixo z paralelo ao eixo de rotação. A origem é posta sobre o eixo z. Não há restrição para onde do eixo ela deve ser estabelecida.

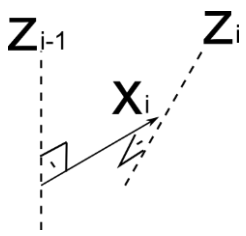
Figura 9 - Eixo Z por DH



Fonte: O Autor (2019).

Em seguida, o eixo x é determinado. Sua orientação é tida como partindo da origem do sistema e apontando para o eixo z da junta seguinte. Por fim, o eixo y é obtido segundo a regra da mão direita.

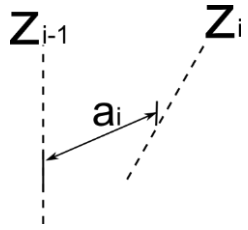
Figura 10 - Eixo X por DH



Fonte: O Autor (2019).

Dessa configuração, quatro quantidades podem ser obtidas e que são suficientes para definir a matriz de transformação homogênea de um elo para o outro. A primeira é o **comprimento da junta (a)**, e é dada como a menor distância entre um eixo z e o seu anterior.

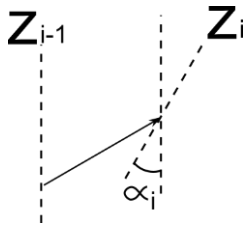
Figura 11 - Comprimento da junta



Fonte: O Autor (2019).

A segunda quantidade é a **torção (α)**. É definida como o ângulo entre o eixo z e a projeção do eixo z anterior.

Figura 12 - Torção

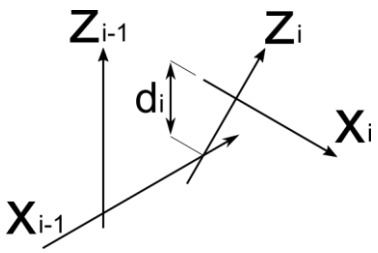


Fonte: O Autor (2019).

Tanto o comprimento de junta quanto a torção são características geométricas de projeto do manipulador. Ou seja, são variáveis próprias do design e constantes.

A terceira variável é o **deslocamento (d)** e pode ser calculada pela distância entre a origem do sistema e a projeção do eixo x do sistema anterior sobre o eixo z.

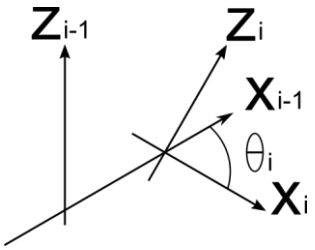
Figura 13 - Deslocamento



Fonte: O Autor (2019).

Por fim, a quarta quantidade é o **ângulo de rotação (θ)**. Ela é definida como o ângulo entre o eixo x do sistema e o eixo x do sistema anterior.

Figura 14 - ângulo de rotação



Fonte: O Autor (2019).

O deslocamento e o ângulo de rotação são chamados de **variáveis de juntas** por estarem associadas ao movimento de sua respectiva junta. Pares prismáticos irão variar o deslocamento enquanto o ângulo de rotação permanece constante, ao passo que juntas de revolução mantém fixo o deslocamento, porém, varia o ângulo de rotação.

Assim, as variáveis necessárias para quantificar cada elo são reduzidas a quatro, das quais três serão fixas e uma estará diretamente relacionada com o grau de liberdade da junta.

2.2.3 Cinemática direta e indireta

Para a realização de tarefas, é necessário o conhecimento da posição e orientação da ferramenta. Na prática, todavia, a manipulação direta dessas variáveis é inviável. Os atuadores agem sobre juntas específicas da cadeia, e o que pode ser efetivamente medido e controlado são os comportamentos dos pares cinemáticos. As configurações do manipulador são atingidas, então através da soma das contribuições de cada junta.

O objetivo da análise cinemática é relacionar as variáveis de junta com a posição e orientação da ferramenta do manipulador. A relação pode ser dada de duas maneiras: a primeira, chamada de **cinemática direta**, recebe as variáveis de junta como entrada e fornece a posição e orientação como saída. A segunda opção é a **cinemática inversa**, que recebe os dados do espaço e fornece as variáveis de junta necessárias para atingir aquela configuração.

2.2.3.1 Cinemática direta

A matriz de transformação homogênea de um par cinemático é constituída a partir da convenção de DH, que utiliza a variável de junta com um dos parâmetros para a definição dos termos da matriz. Em adição, uma vez obtida a matriz, é possível extrair dela as informações de posição e orientação. Isso a torna perfeita para a análise cinemática, pois, recebe a variável e fornece os dados espaciais.

É importante ressaltar que a matriz relaciona uma junta com a consecutiva. Em um manipulador, formado por uma associação de diversos elos, o interesse é extrair a relação entre o elo da base e o atuador. Todavia, havendo conhecimento da matriz para cada par cinemático da cadeia, a relação entre dois pontos arbitários pode ser obtida segunda a e.2.

Supondo um manipulador com n elos, cada um com um grau de liberdade, obtém-se a relação:

$$H_n^0 = H_1^0 H_2^1 \dots H_{n-1}^{n-2} H_n^{n-1}$$

A matriz resultante é da forma:

$$H_n^0 = \begin{bmatrix} n & s & a & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Na qual q é a variável de junta relevante associada a cada elo.

Da matriz pode-se extrair quatro vetores: **n** , **s** e **a** representam para qual direção do espaço a ferramenta está apontando, indicando a variação angular segundo respectivamente os eixos x, y e z. Da última coluna extrai-se o vetor posição, o qual indica em qual ponto do espaço a ferramenta se encontra.

Uma vez que os termos da matriz são funções das variáveis de junta e os vetores citados compostos por tais termos, conclui-se que os vetores são também uma função dessas variáveis. Assim, para cada configuração, pode-se calcular a orientação e a posição da ferramenta no espaço.

2.2.3.2 Cinemática direta do manipulador SCARA

O manipulador SCARA é um dos modelos convencionais e bem comnhecidos na literatura. Pode ser resumido em três juntas: a primeira prismática na altura e duas de rotação no plano horizontal. É conveniente a adição de uma quarta matriz de rotação, referente ao comprimento da ferramenta no último elo da cadeia.

Assim, podem ser definidas as variáveis segundo a convencão de DH, compondo a tabela a seguir

Tabela 1 - Parâmetros de DH

Elo	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	180	0	θ_2
3	0	0	d_3	0
4	0	0	-d_4	0

Fonte: O Autor (2019).

Para o segundo elo, α_2 foi definido como 180 graus pois o referencial foi tomado com braço retraído. Dessa forma, temos para a junta prismática, tem-se:

$$H_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & d_3 \end{bmatrix}$$

Para as demais matrizes de rotação

$$H_1 = \begin{bmatrix} \cos\theta_1 & -sin\theta_1 & 0 & a_1\cos\theta_1 \\ sin\theta_1 & cos\theta_1 & 0 & a_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} \cos\theta_2 & -sin\theta_2 & 0 & a_2\cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Por fim, a ferramenta pode ser considerada como o último cirpo rígido da cadeia, com um comprimento constante na direção z.

$$H_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix}$$

A resultante pode ser calculada multiplicando todas elas na ordem correta

$$H = H_1 H_2 H_3 H_4$$

Da qual se extrai que a posição da extremidade da ferramenta é dada por:

$$x = a_1\cos\theta_1 + a_2\cos(\theta_1 + \theta_2)$$
$$y = a_1\sin\theta_1 + a_2\sin(\theta_1 + \theta_2)$$
$$z = d_3 - d_4$$

Percebe-se, também, pelo vetor da terceira coluna que o eixo z da ferramenta permanece sempre alinhado com o eixo z da base. Essa característica é típica do modelo SCARA. A única rotação ocorre no plano xy, e seu valor é uma composição dos ângulos de

ambas as juntas de rotação. Seu valor, porém, não é de interesse para o processo de fabricação aditiva em questão.

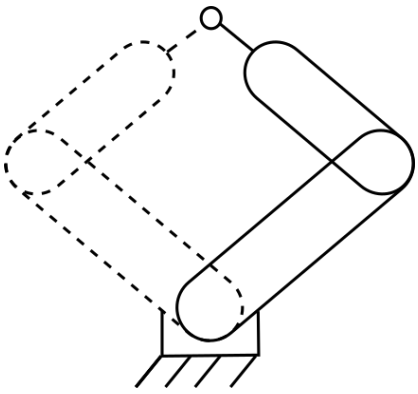
2.2.3.3 Cinemática inversa

Na prática, o que é normalmente conhecido é a configuração do espaço. Dessa forma, deseja-se descobrir qual combinação de posições das juntas resultará no posicionamento adequado da ferramenta. Esse processo de receber as variáveis do espaço e retornar as variáveis de junta é conhecido como cinemática inversa.

Ela é sensivelmente mais complexa que a transformada direta por dois motivos principais. Primeiramente, por seu cálculo não estar explícito na matriz de transformação homogênea, é necessário reorganizar as equações. O resultado é um sistema de equações complexo e frequentemente não linear.

O segundo limitante surge da variedade de soluções que podem ser obtidas. Enquanto a cinemática direta possui exatamente e apenas uma solução, a cinemática inversa pode não possuir nenhuma ou até mesmo múltiplas soluções. Isso se deve ao fato de que é comum aos manipuladores possuírem múltiplas configurações que levem a uma mesma posição e orientação.

Figura 15 - Múltiplas soluções para a mesma posição em um manipulador planar



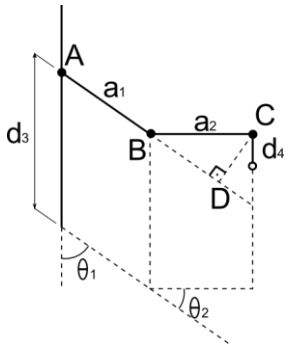
Fonte: O Autor (2019).

O procedimento de cálculo deve ser robusto o suficiente para identificar se aquela configuração no espaço admite solução para o dado manipulador. Em caso afirmativo, deve ser capaz de encontrar e distinguir todas as soluções diferentes possíveis. Para manipuladores mais simples, a solução analítica é viável. Conforme a complexidade da tarefa aumenta, é comum recorrer a métodos numéricos para a solução do problema (ROSÁRIO, 2005).

2.2.3.4 Cinemática inversa para o manipulador SCARA

O manipulador SCARA é suficientemente simples a ponto de possibilitar uma solução geométrica. A configuração pode ser visualizada na figura a seguir.

Figura 16 - Configuração do SCARA



Fonte: O Autor (2019).

Como visto anteriormente, a orientação da ferramenta é sempre paralelo ao eixo z, enquanto que o ângulo de rotação da ferramenta dentro do plano xy é irrelevante para o cenário de estudo. Dessa forma, a análise se limita a posição da ferramenta.

Na direção z, a única junta atuante é a prismática. Somado a influência do comprimento da ferramenta, obtém-se a primeira variável de junta:

$$d_3 = z + d_4$$

Para esse tipo de manipulador, o movimento no plano xy é análogo ao de um manipulador planar de dois graus de liberdade. O ângulo da segunda junta pode ser obtido aplicando a lei dos cossenos no triângulos ABC.

$$cos\theta_2 = \frac{x^2 + y^2 - {a_1}^2 - {a_2}^2}{2a_1a_2}$$

Aplicando a função trigonométrica inversa, o valor da variável pode ser obtido.

$$\theta_2 = arccos\left(\frac{x^2 + y^2 - {a_1}^2 - {a_2}^2}{2a_1a_2}\right)$$

O ângulo da primeira junta é obtido através do triângulo ADC. Uma vez conhecidos os lados, pelo cálculo da tangente, obtem-se:

$$\theta_1 = \arctan(x, y) - \arctan(a_1 + a_2cos\theta_2, a_2sin\theta_2)$$

É válido notar, todavia, que a função arco tangente pode oferecer múltiplas soluções para um mesmo valor dentro do círculo trigonométrico. Isso indica que, para certas

parcelas do volume de trabalho, existe mais de uma configuração associada ao ponto no espaço. Isso pode ser evidenciado geometricamente, como visto na figura 15.

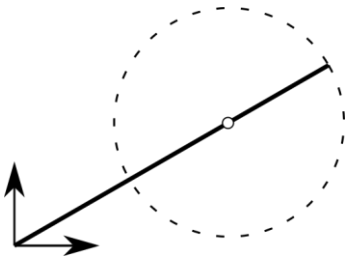
De fato, para todo o interior do espaço alcançável do manipulador, haverão duas soluções possíveis. A exceção está no contorno do volume. Nesses pontos, o braço encontra-se inteiramente esticado ou contraído. Dessa forma, existe apenas uma solução.

2.3 VOLUME DE TRABALHO DO SCARA IDEAL

Primeiramente, é preciso compreender a movimentação do manipulador SCARA ao longo do plano. Uma maneira de facilitar a visualização, sugerida por Rastegar (1990), é realizar o estudo permitindo rotação completa das variáveis de junta. Tanto θ_2 quanto θ_3 poderão rotacionar livremente de 0° até 360°.

Para obter o resultado dessa interação, primeiro será fixada a Junta 2 e a Junta 3 irá rotacionar durante todo seu domínio.

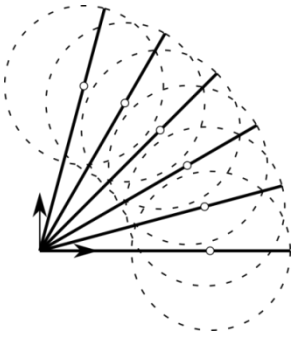
Figura 17 - Alcance movendo apenas a junta 3



Fonte: O Autor (2019).

Essa configuração deverá se repetir para cada movimento da Junta 2. Ao traça-lá para múltiplos ângulos θ_2, obtém-se:

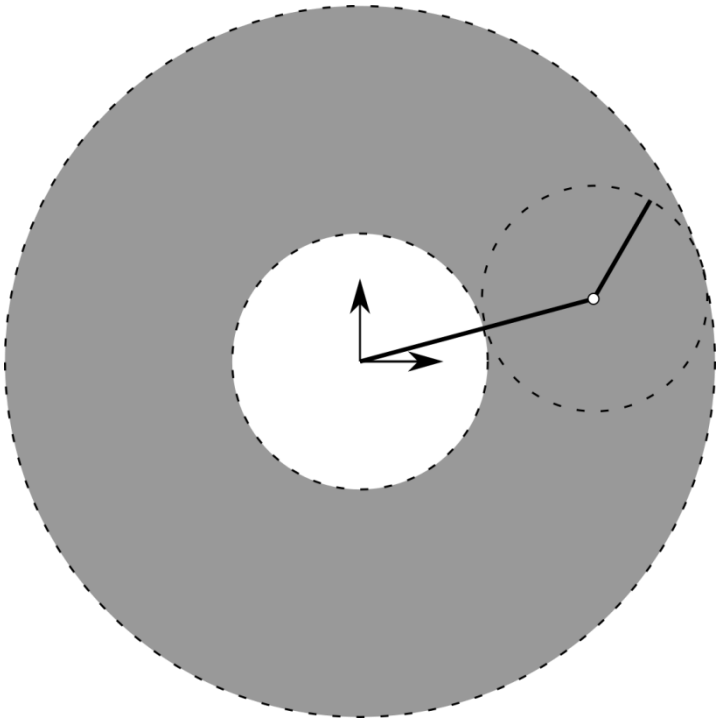
Figura 18 - Alcance para várias configurações



Fonte: O Autor (2019).

Percebe-se que o resultado é um círculo sendo varrido em torno da origem. O lugar geométrico dessa relação nada mais é do que a área entre dois círculos concêntricos centrados na origem.

Figura 19 - Área de trabalho ideal



Fonte: O Autor (2019).

Essa seria a área obtida caso não houvesse limites para as variáveis de junta. Na prática, essas restrições podem existir e a área não possui solução analítica trivial (PATEL, 2015). Todavia, é possui extrair dessa análise os limites mínimos e máximos possíveis para uma configuração qualquer.

2.3.1 Singularidade

Idealmente, para cada ponto alcançável do manipulador, deseja-se que ele seja capaz de desenvolver qualquer velocidade em qualquer direção do espaço. Isso, na prática, não é possível. Existem restrições geométricas construtivas que limitam a movimentação em certas configurações. Um exemplo evidente é quando o manipulador está no limite de seu espaço de trabalho: Apesar de alcançar aquele ponto, não é possível desenvolver velocidades que o levem para fora do volume.

O jacobiano representa a relação entre a velocidade das juntas Q e a velocidade no espaço V.

$$V = J.Q$$

Admitindo V como conhecido, a equação passa a ser um sistema a ser resolvido para se obter o vetor solução Q. Caso esse sistema não admita solução, isso indica que não há um conjunto de velocidades a serem aplicados na junta que irá fornecer a velocidade final desejada na ferramenta. Esse ponto é então uma singularidade.

Matematicamente, o vetor V pode ser interpretado como uma combinação linear das colunas de J.

$$V = J_1 q_1 \quad \cdots \quad J_n q_n$$

Caso pelo menos seis colunas forem linearmente independentes, qualquer velocidade linear ou angular poderá ser alcançada no espaço. Se para uma configuração o número for inferior a seis, haverá alguma restrição ao movimento.

É conveniente citar que, a depender da construção do manipulador, é possível desacoplar o jacobiano de rotação do jacobiano da velocidade angular. A vantagem é a dissociação entre os dois movimentos. Uma singularidade no jacobiano de rotação indica que não é possível assumir alguma orientação. Uma singularidade no outro caso representa uma limitação para transladar em alguma direção específica.

Os pontos que constituem as singularidades podem ser obtidos observando a dependência linear do jacobiano do manipulador. Para o caso de estudo, a orientação do atuador é fixa na direção Z e não varia com as variáveis de junta. Dessa forma, apenas a parcela do Jacobiano referente à velocidade linear é de interesse, podendo ser omitido da análise o Jacobiano da velocidade de rotação.

Sendo composto por três juntas, o jacobiano assume a forma,

$$J = [3\ x\ 3]$$

$$J = [J_1 \quad J_2 \quad J_3]$$

No qual J_i é a componente do Jacobiano associada a junta i. Tal componente assume o valor:

$$J_i = Z_{i-1}$$

Para juntas lineares, na qual Z_{i-1} é o vetor unitário do eixo Z do elo $i-1$. Também

$$J_i = Z_{i-1} \times (o_n - o_{i-1})$$

Para pares de rotação. No qual o_n é a origem do sistema de coordenadas da extremidade do atuador, o_{i-1} é a origem do sistema de coordenadas do elo $i-1$, q_i é a variável de junta e Z_{i-1} é o vetor unitário do eixo Z do elo $i-1$.

Cada valor de J_i obtido corresponderá a um vetor referente à coluna i da matriz Jacobiana. Para o caso em estudo, temos,

$$Z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$Z_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Para as juntas de rotação, é necessário, também, calcular as origens relevantes. Note que as coordenadas da origem de um determinado sistema de coordenadas com relação a um sistema referencial podem ser obtidas através da quarta coluna da matriz de transformação homogênea.

$$H_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_0^2 = H_0^1 H_1^2 = \begin{bmatrix} \cos\theta_2 & -sin\theta_2 & 0 & a_2\cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_0^3 = H_0^2 H_2^3 = \begin{bmatrix} \cos(\theta_2+\theta_3) & -sin(\theta_2+\theta_3) & 0 & a_2\cos\theta_2 + a_3\cos(\theta_2+\theta_3) \\ sin(\theta_2+\theta_3) & cos(\theta_2+\theta_3) & 0 & a_2\sin\theta_2 + a_3\sin(\theta_2+\theta_3) \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Conclui-se

$$o_1 = \begin{bmatrix} 0 \\ 0 \\ d_1 \end{bmatrix}, \qquad o_2 = \begin{bmatrix} a_2\cos\theta_2 \\ a_2\sin\theta_2 \\ d_1 \end{bmatrix}, \qquad o_3 = \begin{bmatrix} a_2\cos\theta_2 + a_3\cos(\theta_2+\theta_3) \\ a_2\sin\theta_2 + a_3\sin(\theta_2+\theta_3) \\ d_1 \end{bmatrix}$$

O cálculo do jacobiano pode então ser realizado,

$$J_2 = Z_1 \times (o_3 - o_1)$$

$$J_2 = \begin{bmatrix} -(a_2\sin\theta_2 + a_3\sin(\theta_2+\theta_3)) \\ a_2\cos\theta_2 + a_3\cos(\theta_2+\theta_3) \\ 0 \end{bmatrix}$$

Analogamente

$$J_3 = Z_2 \times (o_3 - o_2)$$

$$J_2 = \begin{bmatrix} -a_3\sin(\theta_2+\theta_3) \\ a_3\cos(\theta_2+\theta_3) \\ 0 \end{bmatrix}$$

Finalmente

$$J = \begin{bmatrix} 0 & -(a_2 \sin\theta_2 + a_3 \sin(\theta_2 + \theta_3)) & -a_3 \sin(\theta_2 + \theta_3) \\ 0 & a_2 \cos\theta_2 + a_3 \cos(\theta_2 + \theta_3) & a_3 \cos(\theta_2 + \theta_3) \\ 1 & 0 & 0 \end{bmatrix}$$

Em um espaço tridimensional, a matriz precisa de pelo menos três vetores linearmente independentes para assegurar que qualquer velocidade escalar tridimensional seja alcançável. Uma vez que a matriz é 3 x 3, sempre que seu determinante for nulo significa que haverão menos de três vetores linearmente independentes. Assim sendo, as configurações que tornarem nulo o determinante serão singularidades.

$$detJ = \begin{vmatrix} 0 & -(a_2 \sin\theta_2 + a_3 \sin(\theta_2 + \theta_3)) & -a_3 \sin(\theta_2 + \theta_3) \\ 0 & a_2 \cos\theta_2 + a_3 \cos(\theta_2 + \theta_3) & a_3 \cos(\theta_2 + \theta_3) \\ 1 & 0 & 0 \end{vmatrix}$$

$$detJ = a_3 a_2 \sin(\theta_3)$$

$$\sin(\theta_3) = 0$$

$$\theta_3 = n\pi \quad n = 1,2,3 \ldots$$

A singularidade ocorre, então, sempre que a Junta 3 assuma um múltiplo de 180°. Em outras palavras, isso irá ocorrer sempre que o segundo braço do manipulador estiver totalmente esticado ($\theta_3 = 0°$) ou totalmente contraído ($\theta_3 = 180°$). Essas constituem as configurações a serem evitadas em operação.

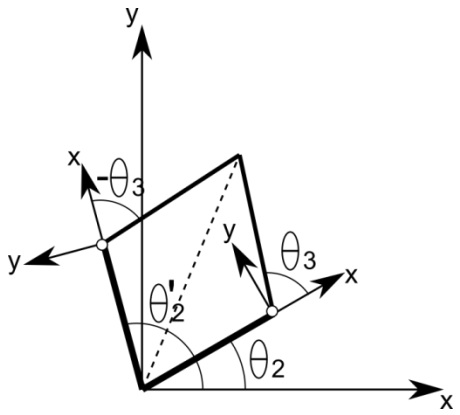
Escolha de configuração

Para uma determinada posição e orientação, o manipulador pode possuir diversas configurações capazes de atendê-las. No âmbito da operação, é necessário escolher critérios para a escolha caso múltiplas soluções se mostrem possíveis. Tais critérios podem incluir a menor trajetória, menor variação das variáveis de junta, menor gasto de energia, etc.

O manipulador SCARA possui a conveniência de oferecer exatamente duas soluções, conforme ilustrado na figura 15. A exceção está nas bordas do volume, quando o braço está totalmente esticado ou totalmente retraído. O critério de escolha resume-se então em definir qual das duas soluções é a mais indicada em um dado momento.

Antes de prosseguir, é importante analisar como ocorrem essas duas configurações e quais as relações entre elas. A figura a seguir explicita a relação das variáveis de junta com o posicionamento.

Figura 20 - Relação entre os ângulos para soluções conjugadas



Fonte: O Autor (2019).

Apesar do ângulo θ_2 ser consideravelmente diferente e não possuír relação aparente, o ângulo θ_3 apresenta um padrão entre as duas soluções. Para uma dada solução da cinemática inversa θ_3, a outra será sempre dada por $\theta_3{}' = -\theta_3$. Isso se torna evidente ao analisar a equação obtida para a cinemática inversa no caítulo 5.

$$\theta_3 = arccos\left(\frac{x^2 + y^2 - {a_2}^2 - {a_3}^2}{2a_3a_2}\right)$$

A função inversa do cosseno é definida no domínio $[0, \pi]$. Expandindo o domínio para uma volta completa $[0, 2\pi]$, passam a existir duas soluções possíveis para a equação: θ_3 e $-\theta_3$. Essas duas soluções irão resultar nas duas configurações possíveis para aquele ponto. Definindo o domínio de θ_3 como $[-\pi, \pi]$, é possível então agrupar as soluções em dois grupos diferentes, um grupo com θ_3 positivo $[0, \pi]$ e outro com θ_3 negativo $[-\pi, 0]$. Cada ponto no espaço irá assumir exatamente uma solução de cada um dos grupos.

Perceba, todavia, que transitar de um desses grupos para outro implica mudar de um domínio no qual θ_3 é negativo para outro no qual ele é positivo e vice-versa. Isso significa que obrigatóriamente será necessário passar por $\theta_3 = 0$ ou por $\theta_3 = \pi$. Porém, esses pontos contituem singularidades e devem ser evitados.

Dessa forma, é prudente evitar a transição de θ_3 de um domínio para outro a menos que seja estritamente necessário. Todavia, cada ponto no espaço possui exatamente uma solução em cada um dos domínios. Logo, é possível representar todos os pontos alcançáveis restrigindo θ_3 apenas a valores positivos ou negativos.

Em um sistema perfeitamente simétrico, não há diferença efetiva entre escolher uma ou outra solução. Entretanto, foi permitido no cálculo da geometria que a mesa estivesse descentralizada com o eixo principal do manipulador. Isso irá resultar que os valores máximos

e mínimos das variáveis de junta ao percorrer toda a área desejada irão mudar. Assim, segundo os critérios estabelecidos, a configuração escolhida será aquela que minimizará a variação angular total das juntas.

2.4 GERAÇÃO DE TRAJETÓRIA

A movimentação do manipulador é mais complexa do que ir do ponto A para o ponto B em linha reta. A maior parte das aplicações requerem movimentos complexos, com trajetórias diversas. A fabricação aditiva é um desses exemplos. Formas das mais diversas e trabalhadas devem ser capazes de serem obtidas, e isso está intrinsecamente ligado à capacidade do manipulador de executar uma enorme gama de caminhos.

A geração da trajetória não se restringe apenas as posições assumidas pela ferramenta. É preciso definir, também, o perfil de velocidade e aceleração desenvolvidos em cada instante. Mudanças abruptas na velocidade levam a picos de aceleração, podendo elevar os torques e forças além do limite dos atuadores. Somado a isso, descontinuidades na aceleração levam a choques e picos energéticos danosos ao sistema.

Há vários métodos que podem ser implementados para sanar esse problema. Merece destaque o sistema de *waypoints*. Esse método consiste na discretização do percurso em diversos pontos chaves, chamados de *waypoints*. Eles são então ligados de maneira consecutiva através de curvas bem definidas selecionadas para facilitar a movimentação do manipulador.

Os métodos de fabricação aditiva *open source* utilizam um mecanismo similar. No lugar de pontos discretos, é gerado um algoritmo em uma linguagem de comando de máquina universal, normalmente o *g code*. O processo é realizado por uma categoria de programas denominados de *slicers*.

Esses programas recebem esse nome devido ao seu modo de funcionamento. Sua função é particionar uma geometria tridimensional qualquer em um conjunto de camadas planas de espessura predefinida. De posse da sessão transversal, é gerado então o conjunto de comandos na linguagem de máquina que irá fornecer o contorno da figura e seu preenchimento.

Conhecidos os comandos a serem seguidos pela ferramenta, é preciso transformá-los em comandos para as variáveis de junta. Nessa etapa, alguns cuidados devem ser tomados. As singularidades devem ser evitadas ao máximo. Nesses pontos, as velocidades passíveis de serem alcançadas são limitadas. Isso implica na possibilidade do manipulador não conseguir

desenvolver o movimento de um ponto para o outro. É preciso mapear o lugar geométrico das singularidades e dispor de mecanismos a evitar tais pontos.

Por fim, a maior parte do volume de trabalho possui mais de uma solução possível. Assim, o sistema deve ser capaz de escolher a mais adequada para o percurso a ser desenvolvido. Um critério deve ser estabelecido para nortear a escolha. Normalmente, a minimização da variação das velocidades de juntas é o critério escolhido (SCIAVICCO, 2006). Porém, evitar as singularidades também é importante.

3 METODOLOGIA

O desenvolvimento do algorítimo de simulação foi divido em duas etapas. A primeira consistiu em estabelecer as dimensões geométricas dos manipuladores, enquanto que a segunda consistiu em produzir um algoritmo de coordenação de movimento.

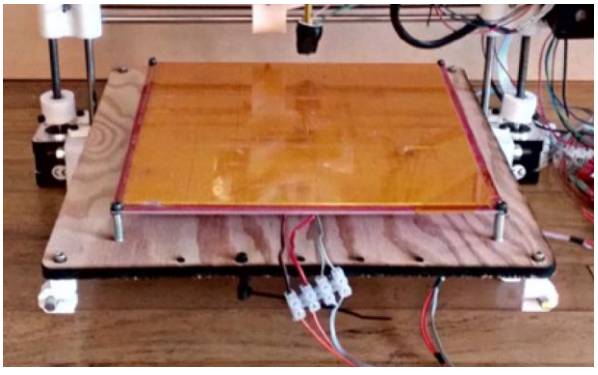
3.1 DIMENSIONAMENTO GEOMÉTRICO DO MANIPULADOR

O estudo do espaço de trabalho de um manipulador, a respeito do que tange as variáveis de Denavit-Hartenberg, é um procedimento intrinsecamente geométrico, como afirma Chernouko (1994). Em outras palavras, a definição das dimensões principais dos elos precede qualquer análise dinâmica e requer apenas o conhecimento das configurações do manipulador em função de suas variáveis de DH e do espaço de trabalho desejado.

Toma-se como ponto de partida, então, a definição do espaço de trabalho desejado para o manipulador. Ao se tratar do processo de fabricação aditiva por depósito de filamento fundido, existe um padrão bem aceito pela comunidade para o espaço de impressão. Esse volume veio associado aos primeiros modelos da prusa e se popularizou com a padronização do mercado de peças para impressoras 3D domésticas.

Tal volume constitui um cubo com 200mm de aresta. Essas dimensões são reforçadas pelo tamanho padrão das mesas de aquecimento, projetadas para possuir uma área útil de impressão de 40000mm². Rigorosamente, uma mesma área plana pode ter uma altura útil tão grande quando desejável. Todavia, o volume cúbico foi o que se popularizou historicamente.

Figura 21 - Mesa para impressão



Fonte: HORVATH (2014).

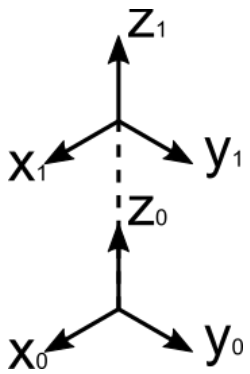
Dessa forma, o volume de trabalho almejado deverá, no mínimo, englobar um cubo contínuo de 200mm de aresta. Isso implica afirmar que o manipulador, a partir de qualquer configuração, possa alcançar um ponto qualquer do cubo sem restrições.

Transformada cinemática do manipulador

A análise das configurações é facilitada sabendo a transformada homogênea de cada elo. Utilizando-se os preceitos definidos no capítulo 5, a primeira etapa é atribuir à cada junta seu sistema de coordenadas e consequentemente suas varáveis de Denavit-Hartenberg.

A primeira junta, nomeada de Junta 1, será o par prismático. Seu eixo Z_1, bem como o eixo X_1 e Y_1, serão paralelos aos da base. Consequentemente, os ângulos α_1 e θ_1 serão nulos. A origem do sistema será tomada a uma distância d_1 da origem absoluta e será a variável de junta. A origem será coincidente ao eixo Z_0 absoluto, o que faz a variável a_1 também nula.

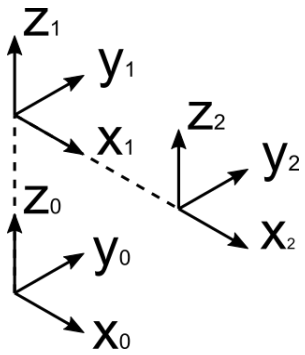
Figura 22 - Sistemas de coordenadas absoluto e da Junta 1



Fonte: O Autor (2019).

O segundo par cinemático, chamado de Junta 2, será o par de rotação acoplado diretamente à junta prismática. O eixo Z_2 será paralelo ao eixo Z_1, distando de a_2 do outro. Por conseguinte, α_1 é nulo. O eixo X_2 é inicialmente paralelo a X_1, e o ângulo θ_2 entre eles é o ângulo de junta. A origem será posta coincidente ao eixo X_1 para que d_2 sejá nulo, simplificando a matriz homogênea.

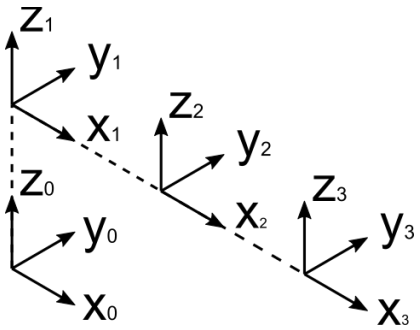
Figura 23 - Sistema de coordenada da junta 2



Fonte: O Autor (2019).

Por fim, a terceira junta, chamada de Junta 3, também de rotação, terá seu eixo Z_3 paralelo a Z_2 e estarão a uma distância a_2 uns dos outros. Analogamente, X_3 tem posição inicial paralela a X_2, com o ângulo θ_3 entre eles sendo o ângulo de junta. A origem do sistema será posta ao longo do eixo X_2. Assim, temos d_1 e a_2 iguais a zero.

Figura 24 - Sistema de coordenada da junta 3



Fonte: O Autor (2019).

Obtém-se finalmente:

Tabela 2 - Variáveis de DH

Elo	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1
2	a_2	0	0	θ_2
3	0	0	d_3	0

Fonte: O Autor (2019).

Seguindo as etapas descritas no capítulo 2, Obtém-se que as coordenadas da extremidade são obtidas pelo conjunto de equações:

$$x = a_2 \cos\theta_2 + a_3 \cos(\theta_2 + \theta_3)$$
$$y = a_2 \sin\theta_2 + a_3 \sin(\theta_2 + \theta_3)$$
$$z = d_1$$

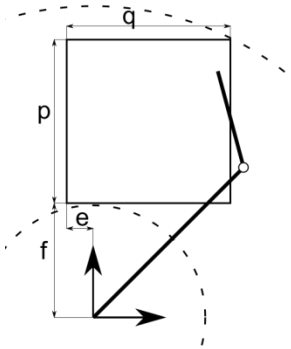
Percebe-se que o movimento vertical depende apenas da variável d_1, enquanto que o movimento no plano XY é uma função de θ_2 e θ_3 apenas. Isso implica que o posicionamento vertical pode ser totalmente desacoplado do movimento no plano xy. A análise das condições limites pode ser realizada de maneira separada para a coordenada Z e para o plano XY.

Análise da área de trabalho ideal.

A partir da transformada cinemática direta conclui-se que o valor máximo na coordenada Z atingido é definido exclusivamente pelo deslocamento máximo na junta linear. Sendo assim, no sentido vertical, basta definir o limite do par cinemático superior à altura máxima desejada. No plano XY, todavia, essa análise não é trivial.

Seja a origem do manipular a mesma do sistema. A mesa estará situada paralela ao plano XY. Ela terá p de altura e q de largura. Sua posição relativa pode ser descrita através do canto mais próximo à origem, distando de e ao longo de X e de f ao longo do eixo Y.

Figura 25 - Configuração da mesa

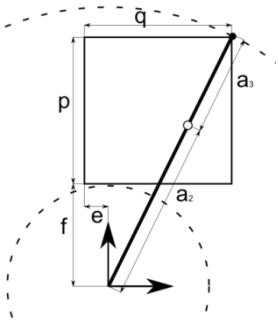


Fonte: O Autor (2019).

Note que a mesa foi descrita com as arestas paralelas aos eixos principais. Essa configuração é, a princípio, arbitrária. Todavia, devido à simetria radial dos limites da área de trabalho, o sistema de coordenadas poderia ser rotacionado sem afetar as propriedades do sistema. O mesmo raciocínio é válido a respeito do quadrante no qual a mesa se encontra. O posicionamento dela à esquerda ou direita de Y, abaixo ou acima de X são análogos devido à simetria.

A partir da figura 25, algumas desigualdades interessantes podem ser calculadas. A primeira a respeito do ponto mais distante com relação à origem, e a segunda relativa ao ponto mínimo.

Figura 26 - Ponto mais distante



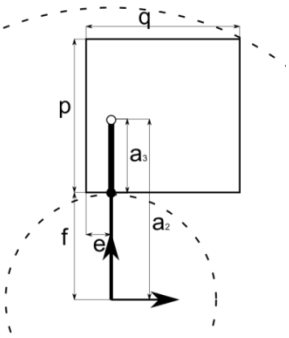
Fonte: O Autor (2019).

Assim,

$$a_2 + a_3 \geq \sqrt{(p-e)^2 + (q+f)^2}$$

Para o mínimo

Figura 27 - Ponto mais próximo



Fonte: O Autor (2019).

Logo,

$$a_2 - a_3 \leq f$$

$$a_3 \geq f - a_2$$

Consequentemente,

$$a_3 \geq \frac{\sqrt{(p-e)^2 + (q+f)^2} - f}{2}$$

O resultado é um valor mínimo do comprimento da Junta 3 para que o manipulador seja capaz de alcançar a extremidade da mesa. O valor máximo será determinado partindo da premissa que todas as peças possam ser impressas pela impressora. Assim seu tamanho máximo deve caber em um retângulo 200mm por 200mm.

$$a_3 \leq \rho\sqrt{p^2 + q^2}$$

Na qual $\sqrt{p^2 + q^2}$ é a diagonal da mesa e ρ é uma constante que regula a fração da mesa tomada pela peça. Seu valor podendo variar de 0 a 1.

O resultado é uma relação para o máximo e mínimo de a_3 que independe da geometria de outras juntas, permitindo seu cálculo individual. a_2 pode ser calculado em conseguinte utilizando as mesmas desigualdades retroativamente. Obtém-se assim

$$a_{3min} = \frac{\sqrt{(p-e)^2 + (q+d)^2} - f}{2}$$

$$a_{3máx} = \rho\sqrt{p^2 + q^2}$$

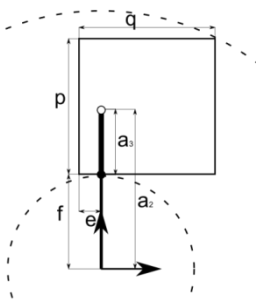
$$a_{2min} = \sqrt{(p-e)^2 + (q+f)^2} - a_3$$

$$a_{3máx} = a_3 - f$$

Existem inúmeras soluções possíveis que satisfazem as equações acima. Foram amostrados diversos pontos até encontrar uma configuração que se adequasse a todos de maneira satisfatória. Os critérios, como descritos anteriormente, foram de minizar os comprimentos das peças bem como as amplitudes das variações angulares. Em adição, foi imposto que as peças fossem menores que o volume de impressão do manipulador.

A configuração obtida está descrita a seguir.

Figura 28 - Configuração do Scara



Fonte: O Autor (2019).

$$a_2 = 190\ mm$$
$$a_3 = 180\ mm$$

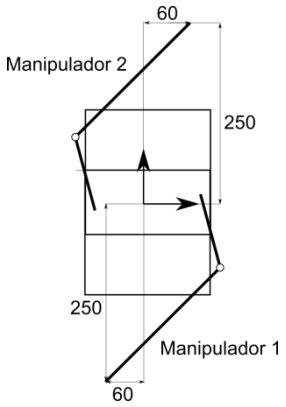
$$e = 40\ mm$$
$$f = 100\ mm$$

Para os ângulos, foi tomado θ_2 variando de 80° até 180°, e θ_3 partindo de -45° até -150°.

3.1.1 Posicionamento multi manipulador

Serão trabalhadas soluções para dois manipuladores, posicionados de maneira oposta um ao outro espelhados segundo o eixo x. Cada manipulador foi posicionado de modo a criar uma intersecção entre as áreas de trabalho. Quanto maior a zona de intersecção, maior será a liberdade de posicionamento das peças. Todavia, menor será o volume total alcançado. Dessa forma, foi escolhida a solução intermediária, sobrepondo metade da bandeja dos manipuladores.

Figura 29 - Sistema multi manipulador com dois SCARA

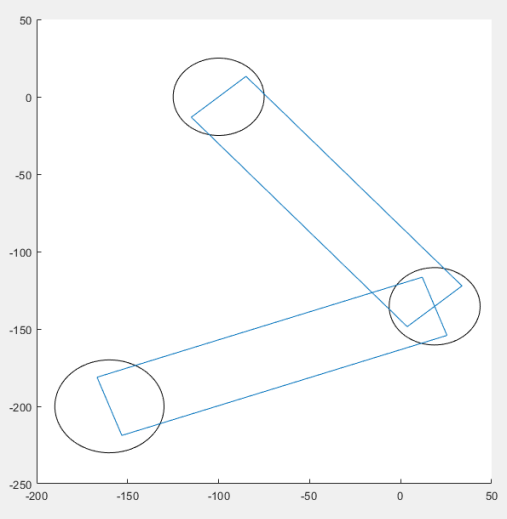


Fonte: O Autor (2019).

3.2 VERIFICAÇÃO DE COLISÃO

O Manipulador pode ser representado pela combinação de círculos e retângulos. A junta do cotovelo e base, bem como a extremidade da ferramenta tem sua representação no plano xy como círculos. Os braços, por sua vez, foram modelados como retângulos. Verificar a colisão entre os robôs se resume a analisar o contato entre esses elementos.

Figura 30 - Representação do manipulador por círculos e retângulos.



Fonte: O Autor (2019).

A função que realiza essa tarefa constrói os manipuladores de maneira gradual. Ela recebe a configuração do manipulador e seus dados construtivos e retorna se há ou não colisão entre algum componente. Primeiramente, o manipulador 1 é construído e posicionado no espaço. O manipulador 2 é colocado em seguida, porém de elemento em elemento. Sempre que uma nova parte é adicionada, é verificada se há intersecção com alguma peça do manipulador.

Devido a forma como são modelados, os tipos de contato se resumem a três. Círculo com Círculo, que matematicamente equivale a calcular a distância entre os centros. Sempre que ela for menor que a soma dos raios, haverá colisão.

Figura 31 - Pseudocódigo para colisão entre dois pontos.

Pseudocódigo para colisão ponto e ponto

Ponto 1 ← Centro da extremidade da ferramenta do manipulador 1
Ponto 2 ← Centro da extremidade da ferramenta do manipulador 2

Se Distância do Ponto 1 ao Ponto 2 $\leq$ Raio$_{ferramenta\ 1}$ + Raio$_{ferramenta\ 2}$
    Colisão ← **Verdadeiro**
Senão
    Colisão ← **Falso**
Fim-se

Fonte: O Autor (2019).

O segundo tipo é entre círculo e segmento de reta. De maneira análoga, consiste em calcular a distância entre ponto e reta, porém com a etapa adicional de verificar se o ponto projeto está dentro dos limites do segmento.

Figura 32 - Pseudocódigo para colisão entre ponto e reta.

Pseudocódigo para colisão ponto e reta

$\overrightarrow{\text{Ponto 1}}$ ← Coordenadas do centro do círculo 1
$\overrightarrow{\text{Ponto 2A}}$ ← Coordenada da primeira extremidade do segmento de reta
$\overrightarrow{\text{Ponto 2B}}$ ← Coordenada da segunda extremidade do segmento de reta
Comprimento ← distância do $\overrightarrow{\text{Ponto 2A}}$ para o $\overrightarrow{\text{Ponto 2B}}$

Se Distância do $\overrightarrow{\text{ponto 1}}$ à Reta $\overline{AB}$ ≤ Raio do círculo 1
    $intersecção \leftarrow Projeção\ do\ \overrightarrow{Ponto\ 1}\ na\ reta\ \overline{AB}$
    { Verificar se a intersecção se encontra nos limites do segmento de reta}
    Se Distância da $intersecção\ ao\ \overrightarrow{\text{Ponto 2A}}$ ≤ Comprimento **E**
        Distância da $intersecção\ ao\ \overrightarrow{\text{Ponto 2B}}$ ≤ Comprimento
            Colisão ← **Verdadeiro**
    Fim-se
Senão
    Colisão ← **Falso**
Fim-se

Fonte: O Autor (2019).

O terceiro modo é o contato entre dois segmentos de reta. Este é análogo a calcular o ponto de intersecção entre retas, seguido por uma etapa adicional para verificar se o ponto se encontra dentro dos limites dos segmentos.

Figura 33 - Pseudocódigo para colisão entre dois segmentos de reta.

Pseudocódigo para colisão reta e reta

$\overrightarrow{\text{Ponto 1A}}$ ← Coordenada da primeira extremidade do primeiro segmento de reta
$\overrightarrow{\text{Ponto 1B}}$ ← Coordenada da segunda extremidade do primeiro segmento de reta
$\overrightarrow{\text{Ponto 2A}}$ ← Coordenada da primeira extremidade do segundo segmento de reta
$\overrightarrow{\text{Ponto 2B}}$ ← Coordenada da segunda extremidade do segundo segmento de reta

Comprimento 1 ← distância do $\overrightarrow{\text{Ponto 1A}}$ para o $\overrightarrow{\text{Ponto 1B}}$
Comprimento 2 ← distância do $\overrightarrow{\text{Ponto 2A}}$ para o $\overrightarrow{\text{Ponto 2B}}$
$intersecção$ ← $intersecção\ da\ reta\ 1\ com\ a\ reta\ 2$

{Verificar se a intersecção se encontra nos limites de ambos os segmentos}
Se Distância da $intersecção\ ao\ \overrightarrow{\text{Ponto 1A}}$ ≤ Comprimento 1 **E**
    Distância da $intersecção\ ao\ \overrightarrow{\text{Ponto 1B}}$ ≤ Comprimento 1
        Se Distância da $intersecção\ ao\ \overrightarrow{\text{Ponto 2A}}$ ≤ Comprimento 2 **E**
            Distância da $intersecção\ ao\ \overrightarrow{\text{Ponto 2B}}$ ≤ Comprimento 2
                Colisão ← **Verdadeiro**
        Fim-se
Senão
        Colisão ← **Falso**
Fim-se

Fonte: O Autor (2019).

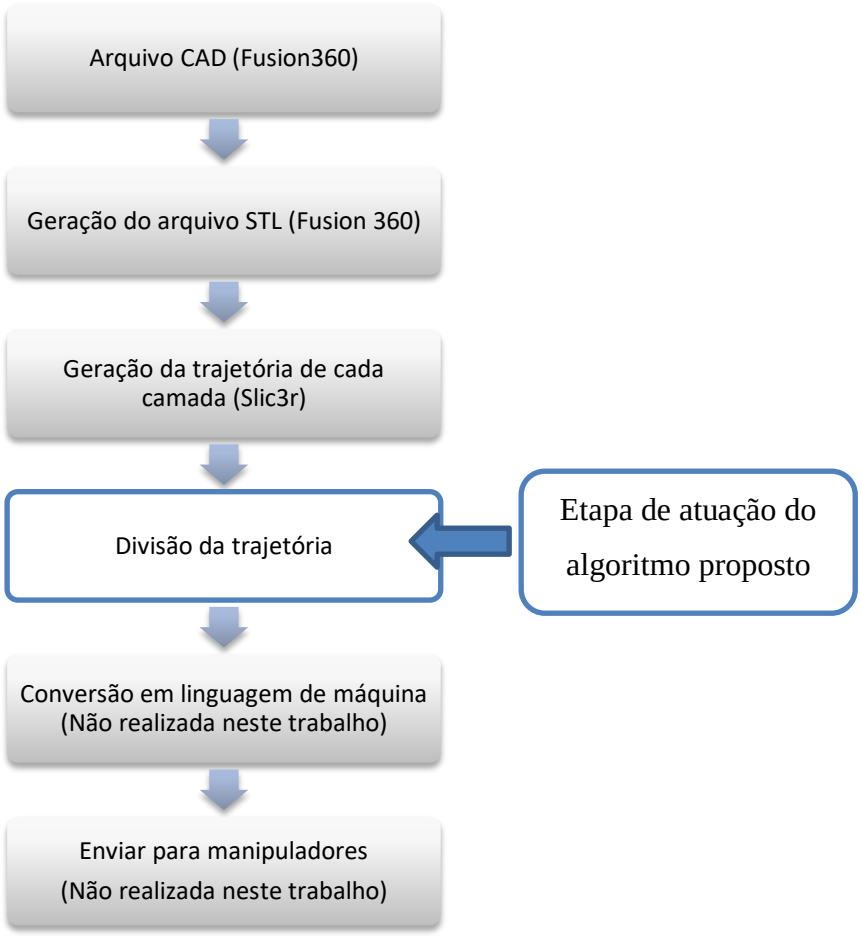
Uma vez definida a trajetória de cada manipulador por completo, é feita uma análise de colisão ao longo do tempo para se certificar que está tudo correto. O tempo é discretizado em um número arbitrário de pontos, que é uma entrada do algoritmo. A cada instante, é recuperada a posição de cada robô e verificado contato. Esse é um método mais assertivo de detecção. Porém é utilizado apenas ao final devido ao alto custo computacional de repartir toda a trajetória.

3.3 COORDENAÇÃO DOS MANIPULADORES

O procedimento de tradução do modelo CAD em linguagem de máquina é relativamente simples. Para cada sessão, o contorno da peça é obtido a partir da interpolação da malha do *STL* . Em seguida o interior do contorno é preenchido com com quaisquer que seja o padrão escolhido (VICENTE, 2016). O resultado é um conjunto de pontos pelos quais o manipulador deve passar e depositar material.

O objetivo do algoritmo é atuar como uma das etapas no processo de conversão do arquivo CAD para a linguagem de máquina. Ele será uma subrotina chamada para cada camada da peça. A entrada será a trajetória para um único manipulador e a saída será a tragetória dividida para cada um dos dois manipuladores.

Figura 34 - Fluxograma do processo de fabricação da peça.



Fonte: O Autor (2019).

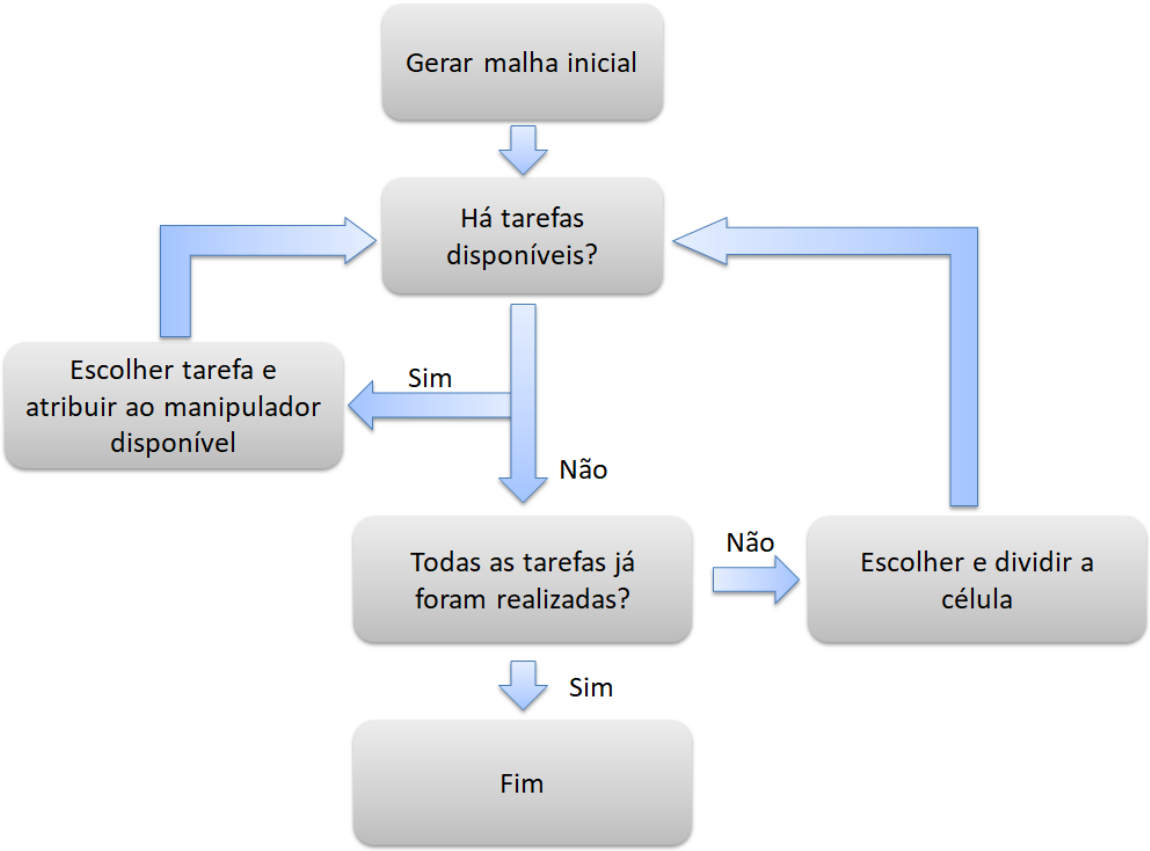
O arquivo CAD é gerado dentro do programa *Fusion360*. Utilizando a mesma plataforma, o arquivo STL é gerado e exportado para o programa *Slic3r*. Dele, é retirado um código em *g-code* com a trajetória das camadas. Esse arquivo é importado como um vetor no *Matlab*, onde os pontos são semparados dos comandos de máquina.

A versão do algoritmo utilizada neste trabalho recebe o conjunto de pontos que compõe o contorno de cada camada da peça. A partir disso, é gerado o preenchimento da peça e a subsequente divisão da trajetória total obtida para cada manipulador.

Figura 35 - Fluxograma do algoritmo implementado.

Fonte: O Autor (2019).

Figura 36 - Fluxograma do algoritmo de divisão.



Fonte: O Autor (2019).

É importante salientar que gerar o preenchimento da peça não é o objetivo do algoritmo mas sim uma escolha de implementação. Como será descrito posteriormente, a trajetória do contorno será tratada de uma maneira diferente da trajetória do preenchimento. Dessa forma, para melhor controlar qual conjunto de pontos pertence ao contorno e qual pertence ao preenchimento, foi optado por implementar a geração do preenchimento.

O desenho das peças foi feito utilizando o *Fusion360*. O contorno das camadas foi gerado a partir do programa gratuito *slic3r*. As trajetórias foram obtidas impondo preenchimento de 0% e apenas 1 camada de contorno. Os valores gerados foram exportados para o *matlab*, no qual o algoritmo foi desenvolvido.

Em um cenário ideal, cada manipulador deveria preencher a peça utilizando o caminho de menor tempo possível. Na prática, a trajetória varia consideravelmente a depender do *Slicer* utilizado. Porém, minimizar o tempo é sempre uma métrica. Ao coordenar múltiplos robôs, todavia, o caminho ótimo individual nem sempre é possível devido ao perigo de colisão. A função do algorítmo de planejamento de movimento é, então, de dividir e organizar

a trajetória de maneira a evitar o contato entre as máquinas mas ainda buscando o tempo mínimo.

Choset (2005) apresenta uma perspectiva para solucionar o planejamento de movimento em sistemas com múltiplos agentes. Nela, os manipuladores são acrescidos de maneira incremental ao sistema. Para cada nova máquina adicionada, a trajetória é calculando considerando os demais como obstáculos móveis. Evidentemente, esse é um método iterativo. As escolhas das trajetórias devem ser ajustada ciclicamente visando encontrar uma solução viável para o problema.

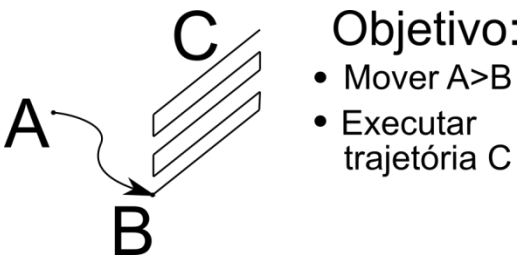
A abordagem aqui utilizada segue um procedimento similar. A cada manipulador será atribuído um conjunto de tarefas a serem realizadas. Sempre que um manipulador terminar uma de suas tarefas, ele irá tentar realizar uma outra ainda não feita. Caso não hajam tarefas disponíveis devido ao posicionamento do outro robô, o sistema volta a algum momento anterior e altera a sequência de tarefas a serem realizadas. O procedimento se repete até uma que uma sequencia viável seja encontrada.

Essa é uma descrição deveras abstrata da solução utilizada. Para torná-la mais concreta, é preciso definir antes dois fundamentos. Primeiramente, faz-se necessário estabelecer quais são, no contexto da impressão 3D, as tarefas a serem realizadas. Em segunda instância, deve-se estabelecer como identificar e evitar a colisão.

3.3.1 Definição das tarefas

Como dito anteriormente, o objetivo é preencher uma determinada área com filamento. Assim, o manipulador deve, primeiro, mover até a região e, uma vez lá, percorrer uma determinada trajetória depositando material.

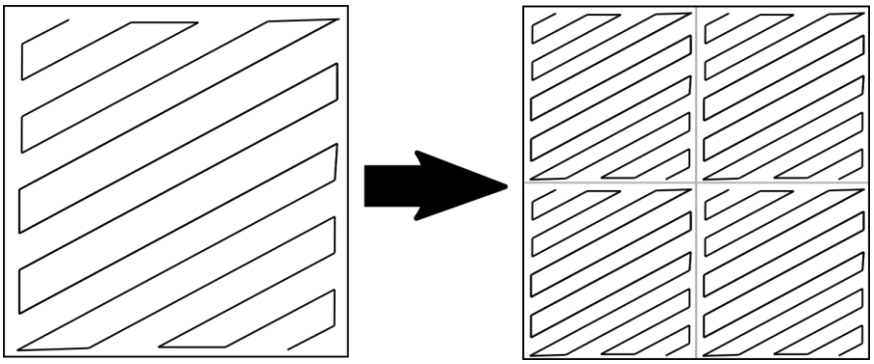
Figura 37 - Definição do objetivo.



Fonte: O Autor (2019).

O preenchimento de uma área pode ser interpretado como uma tarefa a ser executada. A peça pode então ser repartida em um número arbitrário de regiões, cada qual representando um objetivo a ser alcançado pelo robô.

Figura 38 - Divisão da trajetória por área.



Fonte: O Autor (2019).

O resultado disso é a discretização do espaço em uma malha. O problema de sincronização dos agentes é convertido no problema de percorrer cada célula e executar sua tarefa sem que haja colisão. A função do algorítmo é, então, definir como o espaço vai ser dividido e em qual ordem as células serão visitadas.

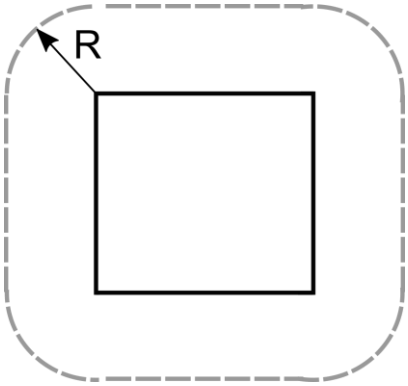
3.3.2 Escolha da célula

Sempre que um manipulador precisar agir, o algorítmo deverá escolher qual tarefa será realizada. A escolha é feita a partir de um critério de pontos. A cada célula é atribuído uma pontuação e será escolhida aquela cujo valor mais se aproxima da diretriz do manipulador. Três critérios são levados em consideração. São eles as células adjacentes, a posição do outro robô e o histórico da célula.

Dos três critérios citados, somente o primeiro é classificatório, sendo os outros dois puramente eliminatórios. O histórico da célula diz respeito a se aquela tarefa já foi realizada ou não. Caso ela já tenha sido visitada, é atribuído um valor simbólico que representa sua indisponibilidade.

Para compreender os outros critérios, é preciso evidenciar como a presença de um manipulador em uma célula influencia as vizinhas.

Figura 39 - Zona alcançada pela cabeça (cinza) e ponta da ferramenta (preto).

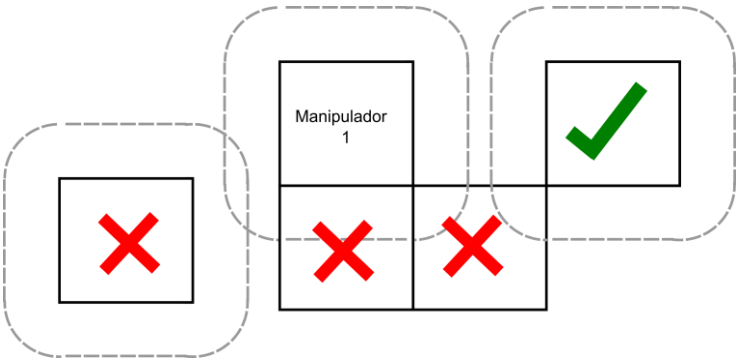


Fonte: O Autor (2019).

Admitindo que todo o perímetro da célula é alcançável, uma distância igual ao raio da extremidade da ferramenta pode ser ocupado pelo corpo do robô. Isso significa que todas as células as quais tenham intersecção com esse apresentam potencial de colisão.

É justo argumentar que essa é uma abordagem conservadora, pois o manipulador não está ocupando toda a região o tempo todo. Todavia, devido a natureza em zigue-zague do preenchimento, a manipulador estará visitando múltiplas fronteiras da célula em diversos instantes de tempo. Assim, a probabilidade de dois manipuladores trabalhando em células vizinhas colidirem é extremamente alta.

Figura 40 - Interferência entre células

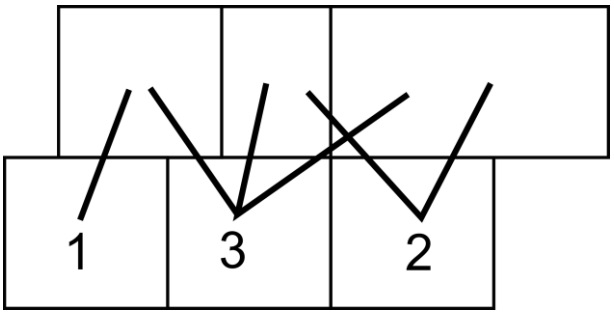


Fonte: O Autor (2019).

A relação de colisão entre as células para o algoritmo é representada por um grafo. Cada célula é um nó. Sempre que uma combinação de nós satisfizer ao critério da distância de colisão, uma ligação é criada entre eles. Ao atribuir os valores às células, o programa verifica a existência dessa ligação. Caso exista, é atribuído um valor simbolizando a indisponibilidade daquela célula naquele momento.

As células restantes são então ordenadas de acordo com sua conectividade. De maneira análoga ao passo anterior, é somado um ponto para cada célula que será inviabilizada enquanto o manipulador trabalha. Cada manipulador possui uma diretriz. Um irá sempre buscar a célula disponível de maior valor, enquanto o outro irá escolher àquela de menor valor.

Figura 41 - Atribuição de valores a células pela seu impacto nas vizinhas



Fonte: O Autor (2019).

A lógica por traz das diretrizes é simples. Um dos manipuladores irá começar nas regiões de maior probabilidade de conflito, visando liberar áreas para o futuro. Enquanto isso, o outro manipulador irá trabalhar nas zonas onde a chance de colisão é mínima, dando tempo para o primeiro realizar suas tarefas. Para os casos de estudo, o manipulador 2, responsável pela metade superior, irá buscar realizar as células de maior conflito, enquanto o manipulador 1, localizado na metade inferior, irá seguir a diretriz oposta.

Figura 42 - Pseudocódigo para escolha das tarefas (continua na próxima página)

Pseudocódigo para escolha das tarefas
Para $i = 1$ **até** $n° \; de \; tarefas$ **faça** $Valor(i) \leftarrow 0$ **Se** $tarefa(i)$ já executada $Valor(i) \leftarrow -1$ **Senão Se** $Colisão \; entre \; tarefa(i) \; e \; tarefa \; do \; outro \; manipulador \geq 1$ $Valor(i) \leftarrow -1$ **Senão** **Para** $j = 1$ **até** $n° \; de \; tarefas$ **faça**

Se $Colisão\ entre\ tarefa(i)\ e\ tarefa(j)\ \geq 1$ ***e*** $tarefa(j)\ não\ executada$

$Valor(i) \leftarrow\ Valor(i) + 1$

Fim-Se

Fim-Para

Fim-Se

Fim-Para

Se manipulador 1

$nº\ da\ tarefa\ \leftarrow$ **Achar** índice do mínimo $Valor \geq 0$

Senão

$nº\ da\ tarefa\ \leftarrow$ **Achar** índice do máximo $Valor \geq 0$

Fim-se

Retorne $Tarefa(n°\ da\ tarefa)$

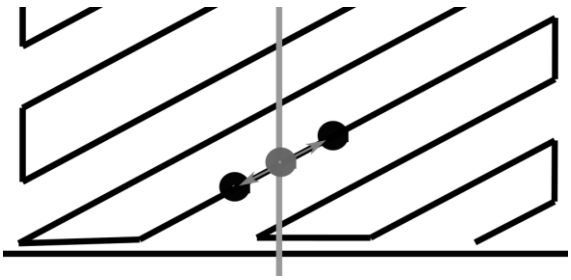
Fonte: O Autor (2019).

3.3.3 Divisão da trajetória

A divisão da célula é, na verdade, a repartição da trajetória em duas. A trajetória é formada por um conjunto de pontos que se combinam dois a dois para gerar uma sequência de segmentos de reta. Para cada segmento, é verificada a intersecção com a reta divisora e os pontos são distribuídos conforme o lado que se encontra da reta.

Caso a reta intercepte o segmento, é criado um ponto de intersecção que irá unir as sequências cortadas. Nesse momento, é importante considerar a espessura do fio depositado. O ponto de intersecção deve ser deslocado a uma distância igual ao diâmetro do filamento. Assim, o preenchimento de cada metade pode ser realizado sem passar por cima da outra metade.

Figura 43 - Offset do ponto de intersecção considerando a espessura do filamento



Fonte: O Autor (2019).

Figura 44 - Pseudocódigo para separação dos caminhos

Pseudocódigo separação dos caminhos

Para $i = 2$ **até** $n°\ de\ pontos\ no\ caminho$ **faça**
Se $x(i)$ e $x(i-1) > x\ de\ corte$
$caminho\ 1\ (fim+1) \leftarrow\ ponto(i)$
Senão Se $x(i)$ e $x(i-1) < x\ de\ corte$
$caminho\ 2\ (fim+1) \leftarrow\ ponto(i)$
Senão
{ Criar ponto de intersecção}
$ponto\ de\ intersecção$ ←intersecção entre a reta $\overline{Ponto(i)Ponto(i-1)}$ e reta de corte
{ compensar diâmetro do extrusor}
$intersecção\ 1\ \leftarrow\ ponto\ de\ intersecção - \overrightarrow{Diâmetro\ extrusor}$
$intersecção\ 2\ \leftarrow\ ponto\ de\ intersecção + \overrightarrow{Diâmetro\ extrusor}$
Se $x(i) < x\ de\ corte$
$caminho\ 1\ (fim+1) \leftarrow\ intersecção\ 1$
$caminho\ 1\ (fim+1) \leftarrow\ ponto(i)$
$caminho\ 2\ (fim+1) \leftarrow\ intersecção\ 2$
Senão
$caminho\ 2\ (fim+1) \leftarrow\ intersecção\ 2$
$caminho\ 2\ (fim+1) \leftarrow\ ponto(i)$
$caminho\ 1\ (fim+1) \leftarrow\ intersecção\ 1$
Fim-se
Fim-se
Fim-Para

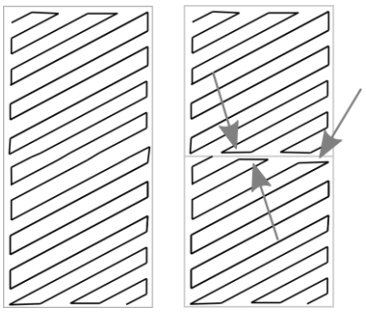
Fonte: O Autor (2019).

3.3.4 Malha inicial

É intuitivo imaginar que quanto menor forem os elementos da malha mais fácil será encontrar a solução. Essa é uma consideração razoável tendo em vista que mais células representão maior flexibilidade no sequenciamento das tarefas e consequentemente menor

chance de colisão. É importante lembrar nessa etapa que as células são subdivisões de uma trajetória, e sua repartição vem com um ônus.

Figura 45 - Caminhos adicionais após repartição



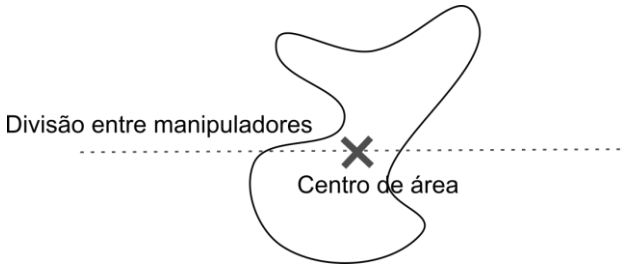
Fonte: O Autor (2019).

A repartição do caminho induz movimentos adicionais à trajetória, aumentando assim o tempo de execução e a quantidade de material depositado. Em adição, as junções entre células são possíveis fontes de defeitos na peça. Consequentemente, busca-se minimizar o número de divisões do espaço.

A solução para esse problema é realizar a repartição da malha dinamicamente. Sempre que o algoritmo encontrar uma configuração sem solução ao sequenciar as tarefas, a malha será subdividida e uma nova iteração terá início. O procedimento continua até o número de divisões ser grande o suficiente para permitir uma solução.

Dessa forma, a malha inicial é consideravelmente simples. A primeira divisão, bastante evidente, separa a peça em duas metades e delimita os movimentos dos manipuladores. Cada um se torna responsável por executar uma das partes. Para que a divisão das tarefas seja o mais simétrica possível, a separação deve ser feita aproximadamente no centro de área da peça. Para os casos de estudo, as peças foram deslocadas de tal forma que seu centro seja aproximadamente coincidente com a origem.

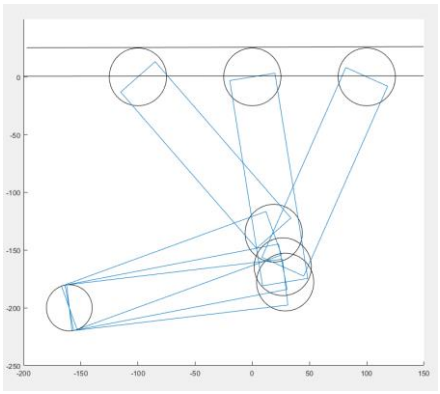
Figura 46 - Centro de área cortado pela reta divisora.



Fonte: O Autor (2019).

A segunda divisão tira proveito de uma característica dos manipuladores SCARA. O posicionamento do seu corpo é bem comportado. Sempre que o mesmo percorre uma reta horizontal, a extremidade do corpo percorrerá uma reta paralela. Isso é verdade sempre que a altura da reta seja maior que o comprimento do braço da base.

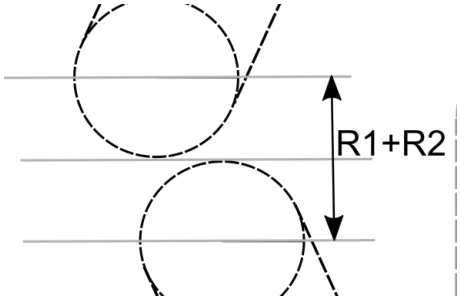
Figura 47 - Comportamento SCARA



Fonte: O Autor (2019).

Levando isso em consideração, pode-se traçar uma reta paralela à primeira a uma distância igual a soma dos raios das extremidades da ferramenta de cada manipulador. Enquanto o robô estiver aquém dessa reta, o perigo de colisão é nulo.

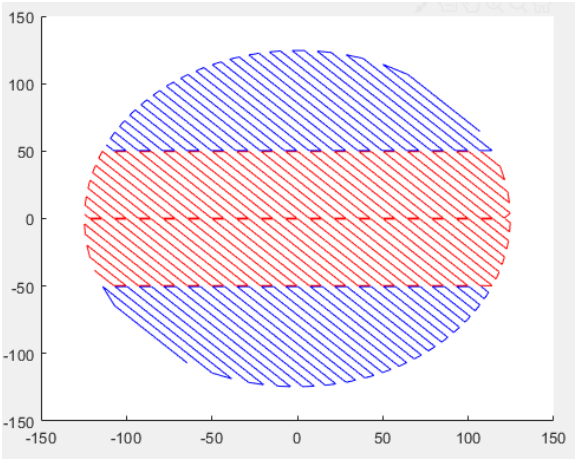
Figura 48 - Zona de colisão entre os manipuladores



Fonte: O Autor (2019).

Assim, a região de cada manipulador é cortada segundo essa distância, totalizando quatro áreas: duas para cada robô. As duas mais próximas do centro apresentam perigo de colisão, enquanto que nas outras duas não há preocupação.

Figura 49 - Malha inicial



Fonte: O Autor (2019).

Essa divisão constitui a malha inicial do problema. Seguindo as diretrizes estabelecidas para os manipuladores, um deles irá tentar executar primeiro a região de colisão de sua metade, enquanto que o outro irá começar pela área mais afastada. Em um cenário ideal, será concluída a região de colisão primeiro. Liberando-a e eliminando quaisquer riscos posterior de colisão. A depender da peça, todavia, o contrário pode ser verdade. Faz-se necessário então repartir a malha e tentar uma nova sequência de tarefas.

Refino dinâmico da malha

Em um dado momento, um manipulador pode precisar agir porém não ter nenhuma célula disponível para ocupar. Nesse contexto, a malha deve ser refinada e uma nova sequência tentada. Antes, porém, é preciso discutir como o tempo progride para o algoritmo.

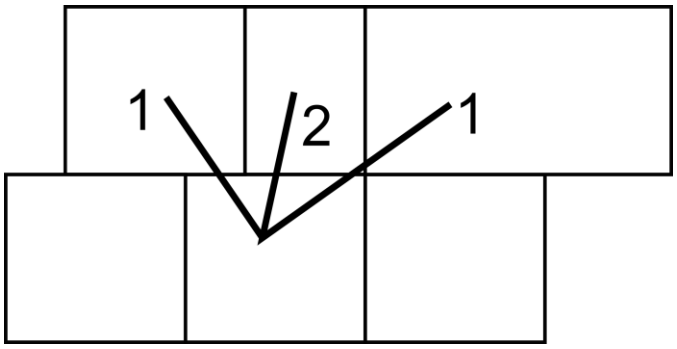
O tempo é discretizado segundo o tempo de permanência nas células. Inicialmente, cada manipulador se encontra em seu ponto inicial, o qual é um input do sistema. Um após o outro, eles escolhem quais tarefas devem executar. Ao defini-las, para cada uma é calculado o tempo de movimento e o tempo de execução. Cada valor obtido representa um marcador na linha de tempo na qual o sistema terá que tomar uma nova decisão.

O tempo então avança até o marcador mais próximo. É registrada a conclusão da tarefa para o respectivo manipulador e uma nova tarefa é escolhida. O ciclo se repete até que ambos os manipuladores tenham concluído todas as suas atividades ou até que não seja possível tomar uma decisão devido a indisponibilidade de tarefas. Para o segundo caso, será necessário retornar o sistema para o ponto anterior à escolha. Dessa forma, antes de realizar uma decisão, o estado do sistema é salvo e associado àquele manipulador.

No caso do sistema não encontrar solução para a malha atual, é preciso decidir qual célula particionar. O conflito surge sempre entre um manipulador querendo mover e o outro durante a execução de uma tarefa. Pelo bem do argumento, tome o exemplo do manipulador 2 estar trabalhando em uma célula e o manipulador 1 buscando uma célula para executar.

A primeira opção é tentar particionar alguma das células alvo do manipulador 1 de tal maneira que uma das metades esteja fora do raio de influência do manipulador 2. A célula candidata seria aquela mais próxima da extremidade do manipulador 2, como mostra a figura 50. No grafo da colisão, comentado anteriormente, as células da vizinhança recebem arestas de peso 1, enquanto que as posteriores recebem peso 2. Para a escolha, o algoritmo toma aquela de valor 1.

Figura 50 - Valores das conexões



Fonte: O Autor (2019).

A segunda opção é realizar a divisão na própria célula do manipulador 2. Dentre as duas, a decisão é feita segundo o tamanho. É repartida àquela de maior área. Células muito pequenas não são desejadas, em parte pelos problemas já citados e em parte devido a perda de informação do padrão do preenchimento devido a sucessivas divisões.

A divisão ocorre no ponto médio da célula segundo o eixo maior da caixa que a contém. Novamente, o objetivo é evitar ter uma malha muito diminuta. O grafo de colisão é atualizado, sendo verificado para cada nó, antes adjacente, se os mesmos permanecem ou não com risco de colisão. O mesmo é feito para o grafo de vizinhança, relevante na etapa de movimento.

Após gerar o novo elemento, é verificado qual foi o manipulador que teve sua tarefa interrompida. O sistema é restaurado para o momento inicial. Ele irá então realizar uma nova escolha e o ciclo continua. O fluxograma abaixo representa o caminhar do algoritmo.

Figura 51 - Pseudocódigo para o avanço do tempo (continua na próxima página)

Pseudocódigo para o avanço do tempo

$Tempo\ 1 \leftarrow 0$

$Tempo\ 2 \leftarrow 0$

Enquanto Houver tarefas não concluídas

{ Verificar qual manipulador terminará a tarefa atual primeiro}

Se $Tempo\ 1 \leq Tempo\ 2$

Se Manipulador 1 já terminou suas tarefas

{esperar manipulador 2}

$Tempo\ 1 = Tempo\ 2 + 0{,}1$

Senão

Escolher tarefa para o manipulador 1

Se Não houver tarefas possíveis

Realisar função de dividir as células

Retornar o sistema ao momento de início

Fim

{ Avançar o tempo}

$Tempo\ 2 = Tempo\ 2 - Tempo\ 1$

{ Definir novo tempo 1}

$Tempo\ 1 = tempo\ de\ movimento + execução\ da\ nova\ tarefa$

Senão

Se Manipulador 2 já terminou suas tarefas

{esperar manipulador 1}

$Tempo\ 2 = Tempo\ 1 + 0{,}1$

Senão

Escolher tarefa para o manipulador 2

Se Não houver tarefas possíveis

Realisar função de dividir as células

Retornar o sistema ao momento de início

Fim

{ Avançar o tempo}

$Tempo\ 1 = Tempo\ 1 - Tempo\ 2$

{ Definir novo tempo 2}

$Tempo\ 2 = tempo\ de\ movimento + execução\ da\ nova\ tarefa$

Fim-Se

Fim

Fonte: O Autor (2019).

3.3.5 Movimento entre tarefas

O deslocamento do manipulador deve tomar em conta o posicionamento do demais para evitar a colisão. Existem algoritmos deveras sofisticados para calcular essa trajetória, como demonstra (CHOSET,2005). O movimento entre tarefas, todavia, representa uma parcela muito pequena do tempo de execução comparado ao tempo dedicado ao preenchimento. Assim sendo, foi tomada uma abordagem simplificada

Existem dois casos a serem avaliados: quando a célula é adjacente e quanto ela não é. No primeiro caso, o movimento é feito de maneira direta. O manipulador vai do ponto atual até a extremidade mais próxima da trajetória da célula. Nesse cenário não há perigo de colisão pois as célula atual e vizinha estão livres.

Para o outro cenário, o manipulador deverá cruzar outras células no caminho que não necessariamente irão estar livres. Para evitar qualquer risco, o manipulador irá, primeiro, se movimentar para fora da zona de colisão, irá se deslocar por fora da zona de colisão até a extremidade da célula alvo e só então se deslocar até o ponto desejado.

3.3.6 Divisão do contorno

A repartição do contorno da peça é uma tarefa um tanto mais complexa. Normalmente, o contorno é feito em um passe único antes de começar o preenchimento. Isso garante uma boa qualidade do acabamento superficial da peça. Nesse contexto, não é interessante utilizar a mesma abordagem de divisão em malha, pois a junção entre as células irá aumentar a rugosidade e reduzir a qualidade. Felizmente, essa etapa tem um impacto consideravelmente menor no tempo de fabricação quando comparado ao preenchimento. Assim, é possível conceder um pouco da eficiência no tempo em troca de minimizar o impacto no acabamento.

Por esses motivos, foi decidido que a trajetória do contorno não seria modificada e seria realizada antes do preenchimento. O contorno foi repartido e cada metade foi atribuída a

um manipulador. As extremidades da trajetória são consideradas como os possíveis pontos de partida. De modo incremental, é verificado se há colisão entre as partes. Caso seja detectada, é testada a combinação oposta para os pontos de partida

Caso nenhuma das combinações dos pontos sejam viáveis, é acrescido atraso do manipulador 2 em relação ao manipulador 1. Esse atraso tem início em 10% do tempo para percorrer a trajetória do manipulador 1. O atraso é aumentado em incrementos de mesmo tamanho até se tornar igual ao tempo de movimento do manipulador 1. Esse seria o pior cenário, no qual um manipulador faz sua parte do contorno e o outro aguarda imóvel.

Ao terminar o movimento, o manipulador 1 dá início à etapa de preenchimento. Uma vez que a diretriz dele é buscar a célula com menor possibilidade de interação, ele irá naturalmente se esquivar do manipulador 2. Assim, o sistema entra na etapa de coordenação do preenchimento, já descrita anteriormente.

Figura 52 - Escolha das tarefas (continua na próxima página)

Pseudocódigo para a divisão do contorno

```
Contorno 1 ← Trajetória da secção do contorno realizada pelo manipulador 1
Contorno 2 ← Trajetória da secção do contorno realizada pelo manipulador 2
Tempo ← Maior tempo de execução entre o contorno 1 e contorno 2
N ← Número de divisões desejadas no tempo
dt ← incremento do tempo
Atraso ← 0

Enquanto Colisão = Verdadeiro
{ Verificar colisão em cada instante de tempo}
    Para i = 1 até N faça
        Calcular posição do manipulador 1 no instante t = i * dt + atraso
        Calcular posição do manipulador 2 no instante t = i * dt
        Se Colisão = Verdadeiro
        { Se ainda não foi feito, testar caso inverso}
            Se Inverter = Falso
                Contorno 2 ← Contorno 2 na sequência inversa
                Inverter ← Verdadeiro
```

```
                    Fim-Se
            { Se já testou o caso invertido, acrescentar atraso}
                    Se Inverter = Verdadeiro
                            Atraso ← Atraso + 10% do Tempo
                            Inverter ← Falso
                    Fim-Se
            { Sair do laço de verificação de colisão}
                    Sair-Para
                Fim-Se
            Fim-Para
Fim
```

Fonte: O Autor (2019).

3.4 CASOS DE ESTUDO

Os casos de estudo irão englobar peças que se beneficiem do grande volume de trabalho permitido pelo sistema. Duas peças serão englobadas. A primeira de geometria simples, dedicada a demonstrar o funcionamento do algoritmo e do impacto das entradas sobre ele. A segunda peça irá avaliar a resiliência do algoritmo ao tentar explorar algumas possíveis fragilidades do mesmo.

Em ambos os casos, as peças terão simetria de extrusão. Ou seja, a secção transversal será a mesma durante todo o eixo Z. Esse perfil foi escolhido, primeiramente, por ser mais simples e adequado para um estudo inicial de viabilidade. Em adição, uma vez que o objetivo é fabricar peças de grandes secções, é naturalmente inviável ter grandes variações do centro de massa. Da mesma forma, as peças precisam ter um tamanho mínimo para que seja viável a operação com múltiplos manipuladores.

Como a simetria é de extrusão, uma vez encontrado uma trajetória viável para os manipuladores em uma camada, ela poderá ser repetida tanto quanto o necessário para atingir a altura desejada. É importante salientar, todavia, que o preenchimento da base e do teto são diferentes do preenchimento do intermédio. Nos extremos, o percentual é de 100% sempre, enquanto que no intermédio esse valor é controlado pelo usuário.

Para os casos de estudo, será considerado um preenchimento de 20%, o qual é coerente com as aplicações médias (HORVATH,2014). O tipo de preenchimento escolhido foi retangular por ser relativamente popular e de considerável simplicidade. Assim, ele deve

ser alternado entre 45º e -45º. Dessa forma, quatro soluções devem ser encontradas para viabilizar a fabricação da peça. São elas 100% de preenchimento com 45º e com -45º, e 20% de preenchimento com 45º e com -45º.

A primeira peça estudada foi uma geometria simples. Consistiu em um círculo de diâmetro 250mm. Essa dimensão é consideravelmente superior ao limite de uma impressora gantry padrão de 200mm x 200mm de área útil. Espera-se que esse caso evidencie o ganho de área do sistema. Em adição, será possível ver o comportamento do algoritmo com certa clareza, dada a simplicidade da peça.

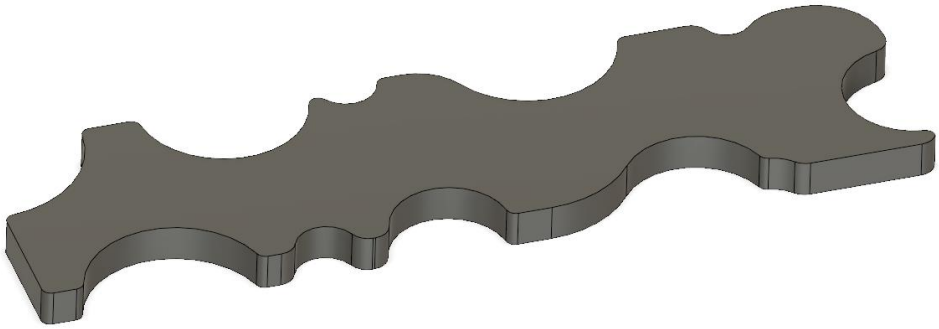
Figura 53 - Peça 1. Modelada em Fusion360

Fonte: O Autor (2019).

A segunda peça tem como objetivo avaliar a resiliência do algoritmo frente a uma peça que impõe uma série de condições adversas. A peça está completamente restrita a área de colisão, dessa forma, sempre haverá risco de contato entre os corpos. A peça é assimétrica, com largura variando consideravelmente e aperiodicamente ao longo do comprimento. Assim, a área de cada célula irá diferir bastante de uma para a outra. Em adição, existe a possibilidade de gerar células vazias, as quais devem ser descartadas da solução.

A peça foi gerada esculpindo um retângulo de 80mm x 240mm. Foram utilizadas medidas arbitrárias para os rasgos realizados na peça.

Figura 54 - Peça 2. Modelada em Fusion360



Fonte: O Autor (2019).

Em ambas as peças, será comparado o tempo de execução estimado antes da divisão e após a divisão entre os manipuladores. O critério de avaliação do tempo será por comprimento total do caminho. Em outras palavras, será somado a distância entre os pontos da trajetória e comprimento total será usado como base comparativa do tempo de execução. Os dados das peças podem ser obtidos a partir do arquivo “peca1.mat” e “peca2.mat” fornecidos em conjunto com esse trabalho.

4 ANÁLISE DOS RESULTADOS

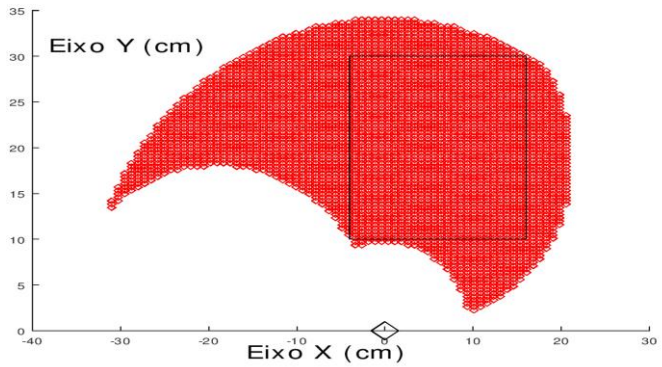
Os resultados foram analisados em dois momentos. O primeiro quanto ao ganho de área individual e conjunta dos manipuladores. Em um segundo momento, foi avaliado os critérios mínimos para o algoritmo de colisão e sua eficiência no tempo.

4.1 VERIFICAÇÃO NUMÉRICA DA ÁREA DE TRABALHO

Com auxílio do programa *matlab*, foi calculado numericamente a área de trabalho no plano XY. Foi gerado um espaço retangular de dimensão $(a_2 + a_3) \times (a_2 + a_3)$ capaz de englobar os limites máximos do manipulador. Tal espaço foi subdividido em n^2 pontos igualmente espaçados. Um algoritmo foi então desenvolvido para estimar numericamente a área de trabalho.

Para cada ponto, o algoritmo realiza a transformada inversa e verifica se o mesmo admite solução. Em caso negativo, aquele ponto não está englobado no espaço de trabalho. Caso contrário, aquele ponto é salvo. Ao fim, o conjunto de pontos obtido é representativo da área alcançável pelo manipulador.

Figura 55 - Relação da área de trabalho do manipulador SCARA (vermelho) e da área desejada (preto)

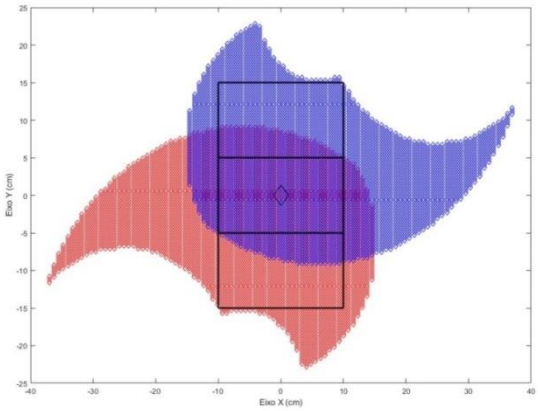


Fonte: O Autor (2019).

Percebe-se claramente que os limites da mesa estão contido na área de trabalho do manipulador. De fato, atribuindo a cada ponto uma fração do espaço, a área total calculada é de aproximadamente 92000mm², mais do que o dobro da área útil da mesa. Isso revela um potencial significativo para expandir o volume de trabalho de maneira considerável.

Ao combinar a área dos dois manipuladores, obtém-se uma área combinada de aproximadamente $1{,}4 * 10^5 mm^2$.

Figura 56 - Combinação da área do manipulador 1 (vermelho) e manipulador 2 (azul). Em preto, as áreas de referência de cada manipulador



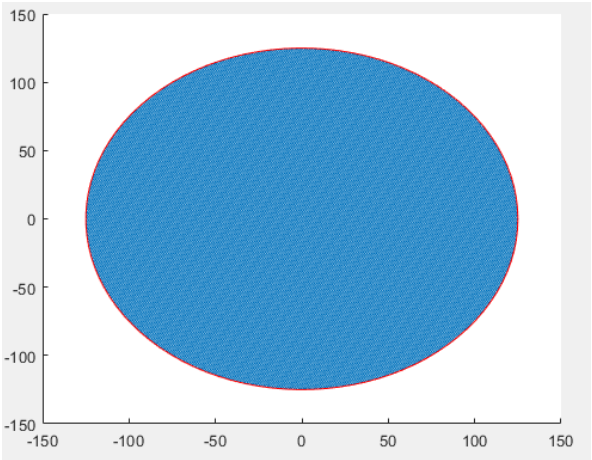
Fonte: O Autor (2019).

4.2 CASOS DE ESTUDO

Foi atribuída uma espessura de 20mm para o braço e diâmetro de 25mm para a extremidade da ferramenta e junta de cotovelo baseando-se nos modelos de Ogulmos (2016) e Okabe (2016).

4.2.1 Caso 1

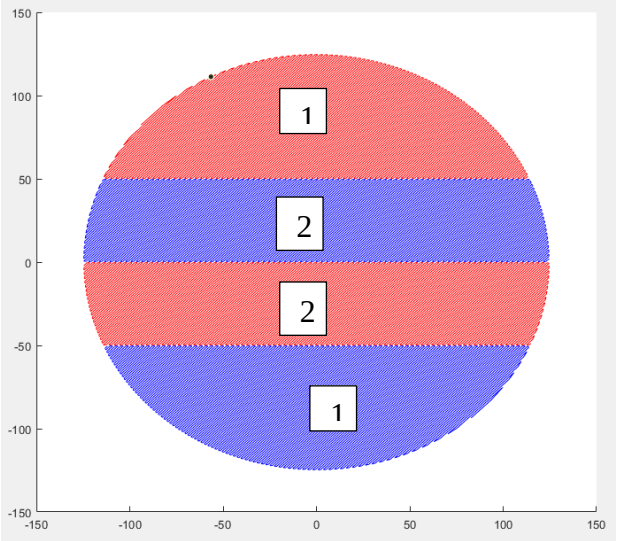
Figura 57 - Peça 1 preenchimento e contorno. Eixos em milímetros



Fonte: O Autor (2019).

A primeira etapa de construção do preenchimento foi executada com sucesso. É possível verificar o espaçamento entre o preenchimento e as curvas do contorno.

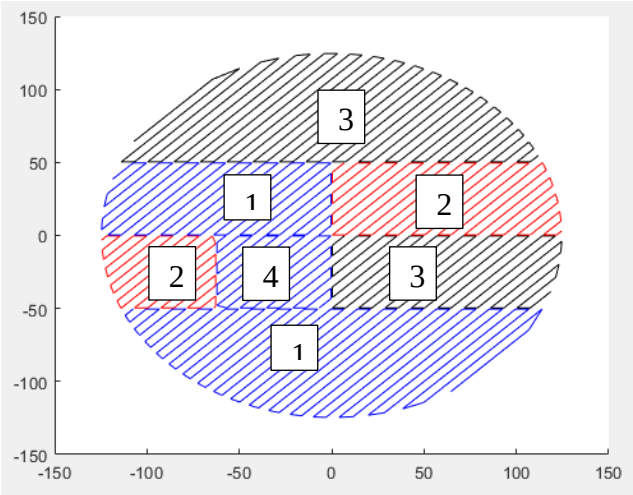
Figura 58 - Peça 1 malha padrão 45º 100%. Eixos em milímetros



Fonte: O Autor (2019).

Observando o sequenciamento das células, percebe-se o comportamento segundo as diretrizes impostas. Um manipulador executa a tarefa mais próxima do centro enquanto o outro toma a mais afastada.

Figura 59 - Peça 1 Malha padrão 45º 20%. Eixos em milímetros

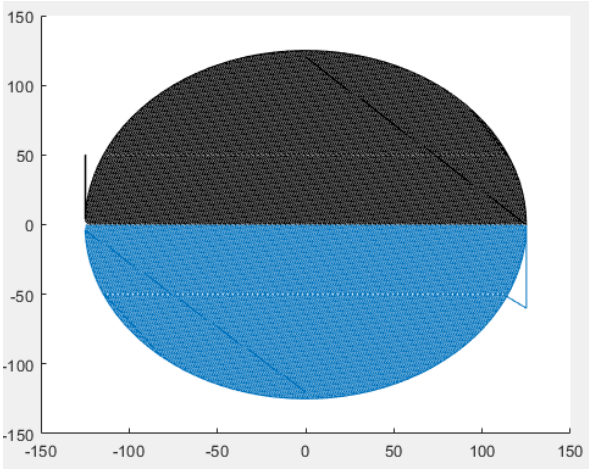


Fonte: O Autor (2019).

Curiosamente, ao reduzir o preenchimento, o manipulador 1 terminou sua primeira tarefa mais cedo. Isso fez com que o algoritmo tivesse que se adaptar e buscar uma nova solução para a trajetória.

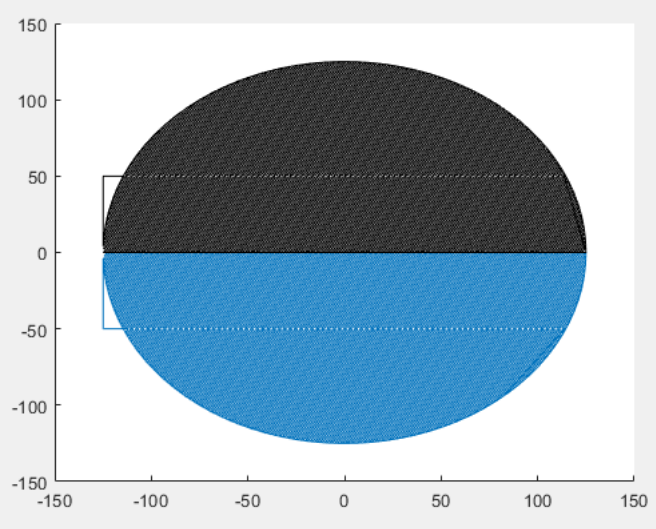
É notável, também, uma inversão na ordem das tarefas 3 e 4 do manipulador 1. Em um caminho ótimo, a célula 4 seria executada antes da célula 3. Essa inversão se deve ao fato que, nesta configuração, o algorítimo vê as duas células com prioridades iguais. Dessa forma, escolhe a primeira da lista de tarefas, que porventura foi a célula mais a direita. A solução para o problema seria impor um critério de desempate adicional, tal qual a distância entre células.

Figura 60 - Peça 1 padrão -45º 100%. Eixos em milímetros



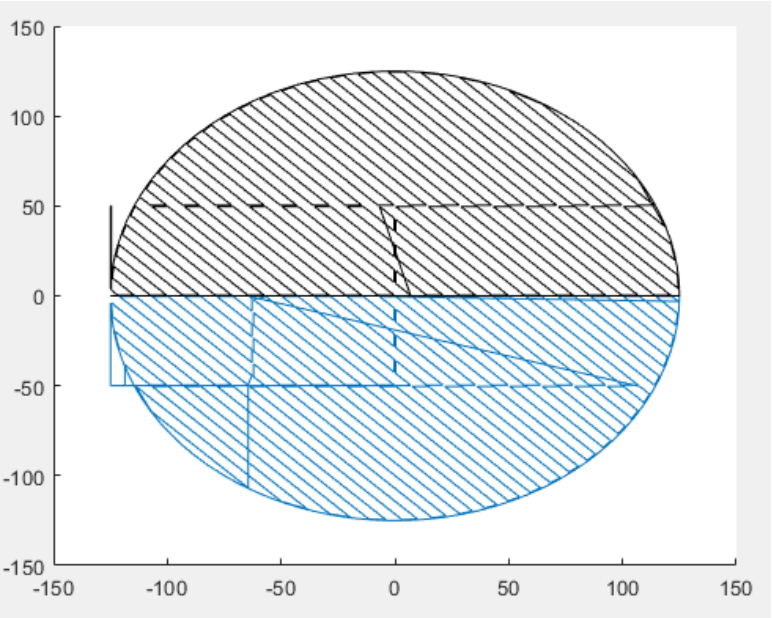
Fonte: O Autor (2019).

Figura 61 - Peça 1 padrão 45º 100%. Eixos em milímetros



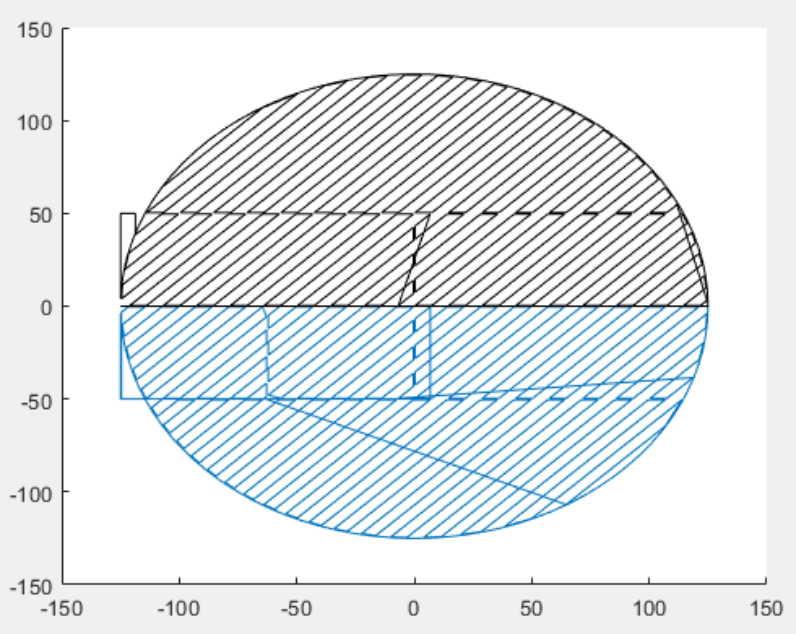
Fonte: O Autor (2019).

Figura 62 - Peça 1 padrão -45º 20%. Eixos em milímetros



Fonte: O Autor (2019).

Figura 63 - Peça 1 padrão 45º 20%. Eixos em milímetros

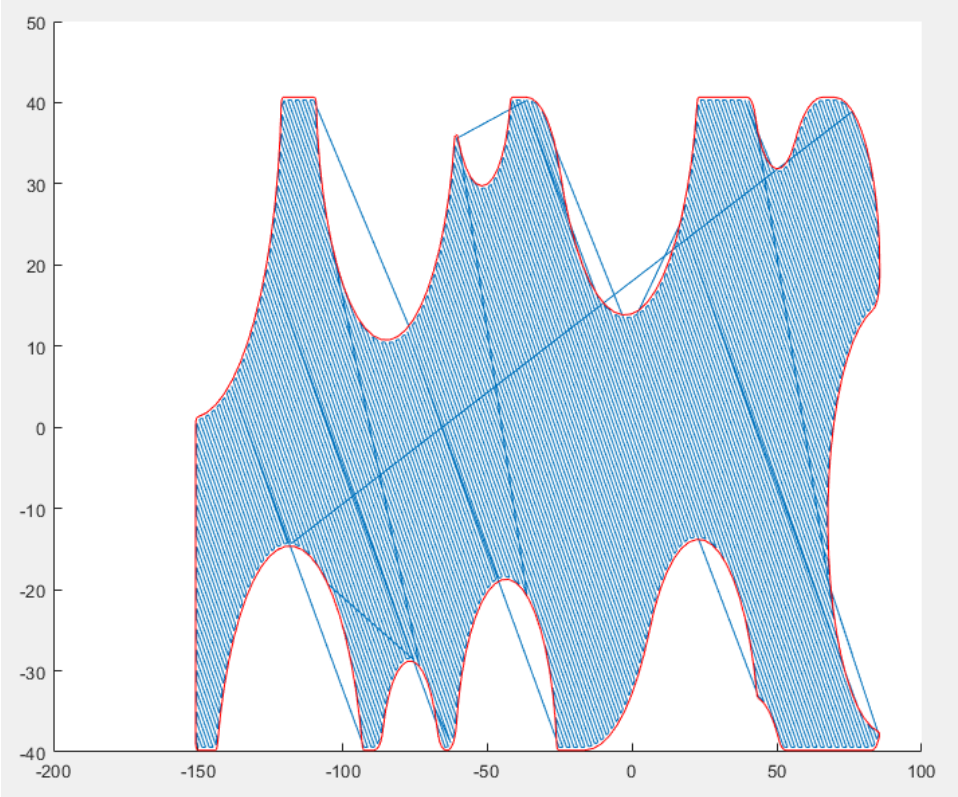


Fonte: O Autor (2019).

Para as demais funções, o resultado é igualmente eficiente. Diferentemente da primeira camada, o ponto inicial das demais foi tido como o último do manipulador. Esse cenário é mais realista, na qual busca-se minimizar o tempo de movimento do manipulador.

4.2.2 Caso 2

Figura 64 - Peça 2 contorno e preenchimento. Eixos em milímetros

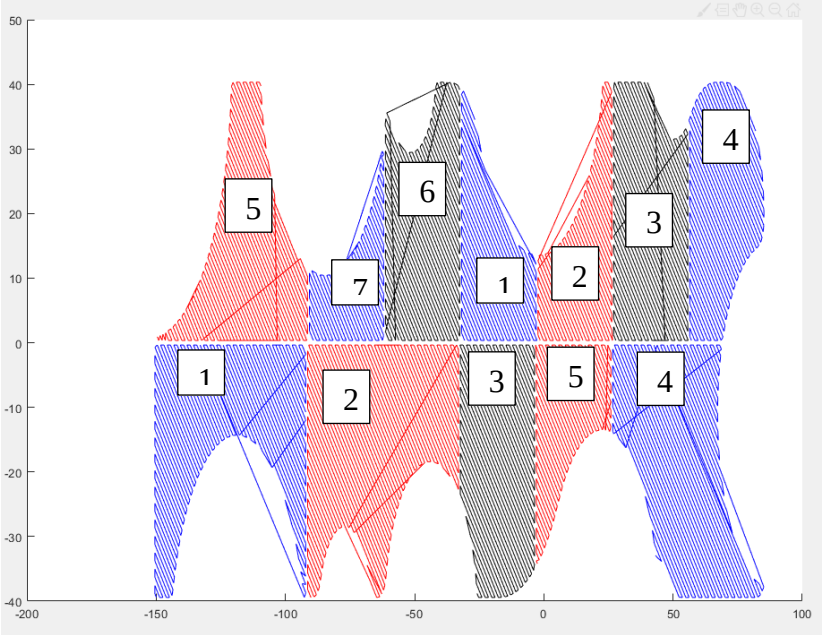


Fonte: O Autor (2019).

O primeiro elemento a se notar são as linhas desordenadas ligando regiões aparentemente aleatórias da peça. Esse fenômeno se deve a uma limitação do *slicer* implementado. Em alguns momentos, a reta descrita pelo preenchimento atravessa o contorno em múltiplos pontos, gerando assim múltiplos segmentos. Esses segmentos precisam ser armazenados separadamente para depois serem unificados. Caso contrário, corre-se o risco ligar dois pontos por fora da peça.

A consequência negativa é que o preenchimento arrisca não ficar perfeitamente ordenado, como no caso anterior. Apesar de uma pequena perda de eficiência no tempo, Isso não é um problema grave para os métodos de depósito de filamentos, que podem interromper a extrusão ao passar por um região onde não devam depositar material.

Figura 65 - Peça 2 100% inclinação 45º malha final. Eixos em milímetros



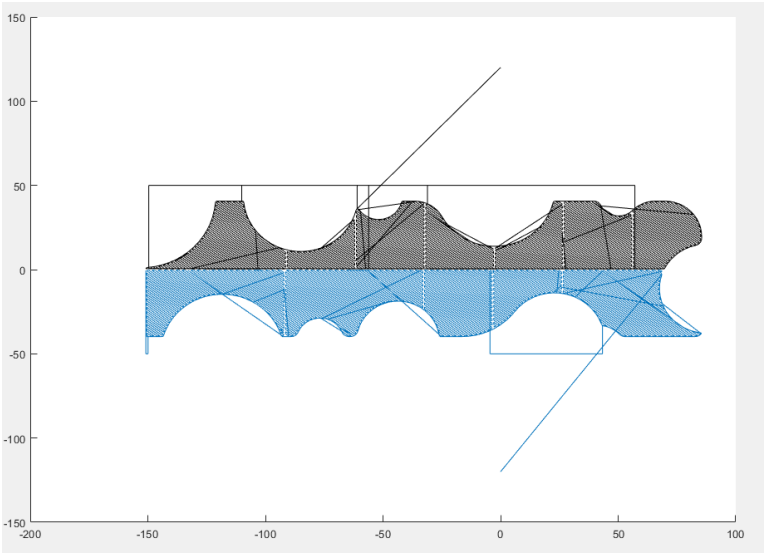
Fonte: O Autor (2019).

A repartição do espaço em células foi, de maneira geral, efetiva. Não foram encontradas grandes discrepâncias de tamanho entre os preenchimentos. De fato, O algoritmo parece tender a uniformizar o tamanho deles. Essa característica pode ser possivelmente aproveitada inserindo um critério enviesado para a repartição da célula, tal qual dividir segundo o centro de área e não o centro da caixa de contenção.

Em contrapartida, algumas células apresentaram uma degeneração do perfil esperado para o preenchimento. É exemplo o célula 1 e 2 do manipulador 1. Esse inconveniente se origina quando a área de um dos lados da divisão é consideravelmente pequena. Para o presente estudo, isso não ocasionou um problema na representação da peça. Porém, caso fosse necessário continuar dividindo a célula, existe a possibilidade de perder informações com o desaparecimento de pontos. Uma possível solução para esse problema é estabelecer um critério de tamanho mínimo de divisão da célula.

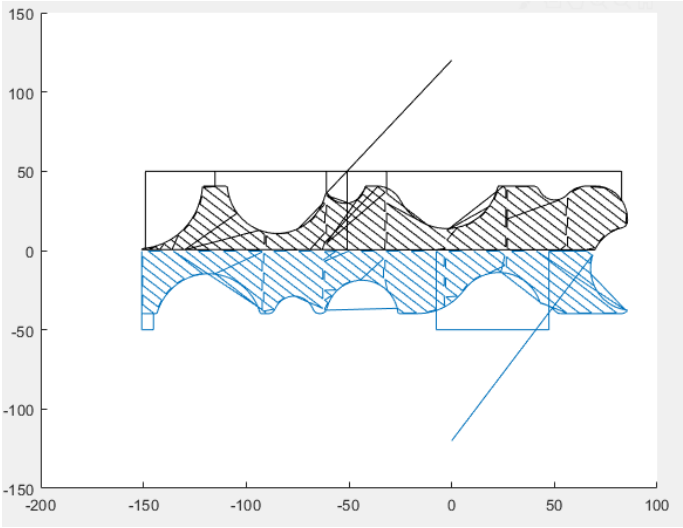
Apesar dos pequenos inconvenientes, o algoritmo foi eficaz em trabalhar a peça, mesmo com todas as características adversas. A análise discreta no tempo não acusou nenhuma colisão entre os manipuladores para nenhum dos quatro casos.

Figura 66 - Peça 2 padrão -45º 100%. Eixos em milímetros



Fonte: O Autor (2019).

Figura 67 - Peça 2 padrão -45º 20%. Eixos em milímetros

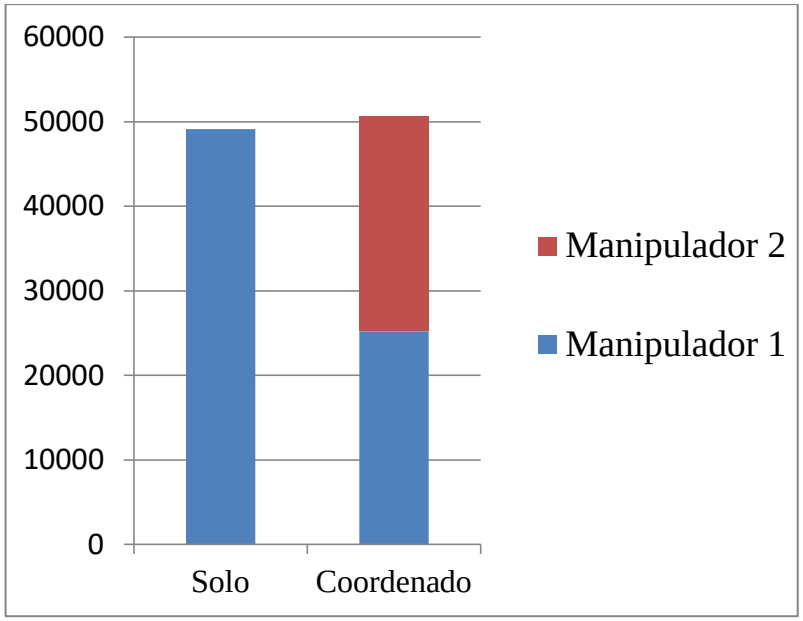


Fonte: O Autor (2019).

4.2.3 Análise do tempo

Para a peça 1 com 100% de preechimento, o comportamento foi próximo ao ideal. De fato, a divisão do tempo de tarefa foi de aproximadamente 51% para cada manipulador, com uma razão de 99,2% entre eles.

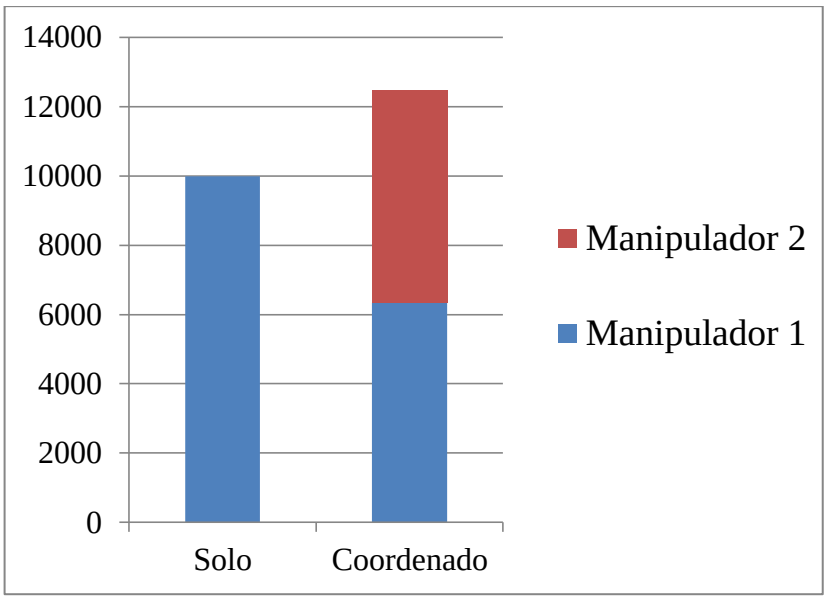
Figura 68 - Tempo de execução da peça 1 45º 100% de infill (mm)



Fonte: O Autor (2019).

No caso da peça 1 com 20% de preenchimento, houveram mais divisões. Como esperado, o aumento do número de células tem impacto negativo na eficiência. O tempo de execução aumentou para 61,6% no manipulador 1 e 63,5% no manipulador 2. A razão de distribuição entre eles foi de 96%.

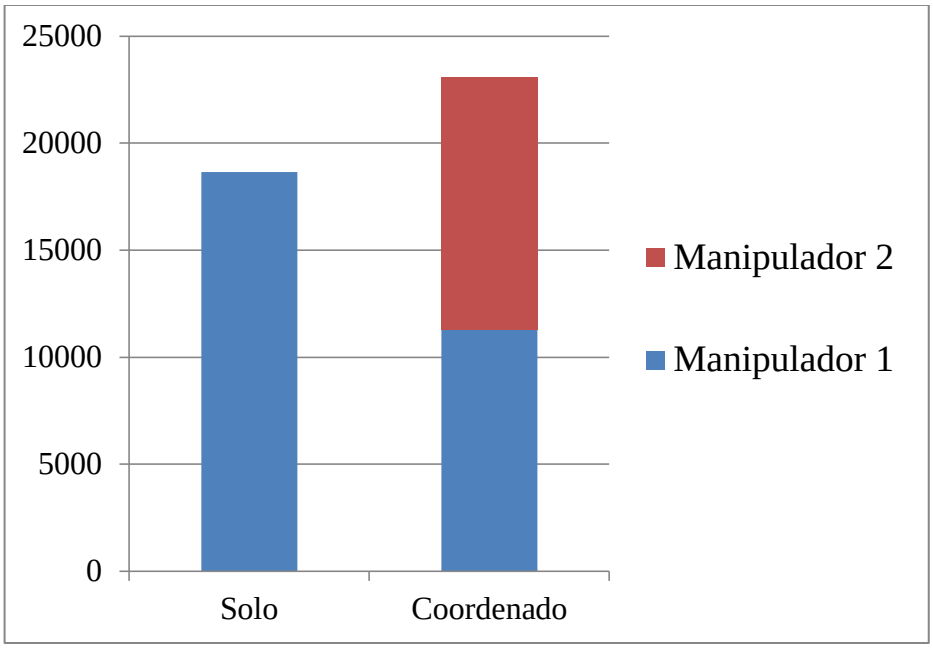
Figura 69 - Tempo de execução da peça 1 45º 20% de infill (mm)



Fonte: O Autor (2019).

Para a peça 2, a qual também conta com uma quantidade considerável de divisões, o resultado foi similar. A fração do manipulador 1 foi de 60,6% aproximadamente enquanto que a do manipulador 2 foi de 63,1%. A razão entre eles foi também de aproximadamente 96%.

Figura 70 - Tempo de execução da peça 2 45º 100% de infill (mm)



Fonte: O Autor (2019).

5 CONCLUSÃO

No presente trabalho, foi dimensionada a configuração geométrica e disposição de um sistema múlti manipulador para aumento do volume de trabalho de impressoras 3D por depósito de filamento fundido. Em adição, foi implementado um algoritmo de cooperação entre os manipuladores para o trabalho simultâneo em uma mesma peça.

O volume de trabalho calculado foi consideravelmente superior àquele das impressoras convencionais mesmo utilizando componentes de tamanhos similares. A partir da combinação das áreas úteis, tornou-se viável a fabricação de peças consideravelmente grandes no contexto da impressão 3D.

O algoritmo utilizado para coordenação de tarefas foi implementado com sucesso no quesito de colisão. O método de divisão dinâmica do espaço acompanhado do critério de escolha eficiente permitiu repartir o espaço em um número mínimo de elementos, mantendo assim a integridade e eficiência do preenchimento. Mesmo diante de contornos complexos e desordenados foi possível encontrar uma trajetória viável.

Mesmo nos piores cenários exemplificados, a redução do tempo total foi bastante relevante. Uma queda de 40% tem grande impacto considerando que os processos requerem muitas horas de operação. A sincronia dos manipuladores se manteve alta, com uma razão próxima a 95%. Isso indica não há grandes perdas de eficiência ao realizar a sincronização entre camadas.

Uma das maiores limitações para o sistema implementado foi o Slicer. Ao se deparar com curva côncavas ou corpo com rasgos complexos, o preenchimento obtido não foi o ideal. A implementação de um software mais sofisticado ou até mesmo a compatibilidade com um Slicer já existente pode melhorar a qualidade da solução obtida.

O bom funcionamento do algoritmo os casos de estudo abre a perspectiva de novos casos nos quais as restrições presentes nestes sejam gradualmente revogadas. Limitações tais quais secções variáveis e peças com centro de área variante ao longo das camadas. Diversas aplicações podem ser estudadas a partir da sofisticação do método aqui proposto.

Desta forma, este trabalho cumpriu seu objetivo de utilizar a configuração SCARA em um sistema multi manipulador aplicado a impressão 3D por depósito de filamento fundido. Os resultados obtidos foram viáveis para as peças propostas e se mostraram promissores a serem expandidos para peças mais complexas.

REFERÊNCIAS

BELLINI, A. Liquefier dynamics in fused deposition. **Journal of Manufacturing and Engineering**, v. 126, 2004.

CASTELLI. G.; OTTAVIANO. E. Manipulators workspace analysis as based on a numerical approach: theory and applications. In:_____. **Manipulators***:* theory, types and applications. Nova Iorque: Nova Science Publisher, 2011.

CHERNOUKO. F. L. BOLOTNIK. N. N.; GRADETSKY. G. V. **Manipulation robots: dynamics, control and optimization**. Florida: CRC Press, 1994.

CHOSET. H. et al. **Principles of robot motion:** theory, algorithms, and implementations**.** Boston: MIT Press, 2005.

CRAIG. John. J, **Introduction to robotics**: mechanics and control. 2. ed. Nova Jersey: Pearson Education, 1989.

CUNHA. L. B. **Elementos de máquinas**. Rio de Janeiro: LTC, 2005.

GHOSAL, A. **Robotics**: fundamental concepts and analysis. Oxford: Oxford Higher Education, 2006.

HORVATH, John. **Mastering 3d printing**. California: Apress, 2014.

KAMRAN, M.; SAXENA. A. A comprehensive study on 3D printing technology. **MIT International Journal of Mechanical Engineering**, v. 6, n. 2, p. 63-69, 2016. ISSN 2230-7680.

LYNCH. K. M.; PARK. F. C. **Modern robotics:** mechanics, planning, and control. Cambridge*:* Cambridge University Press, 2017.

NORTON. R. L. **Cinemática e dinâmica dos mecanismos**. São Paulo: Mc Graw Hill, 2010.

OGULMUS. S.; ÇAKAN. A.; MUSTAFA. T. A. Modeling and position control of Scara type 3D printer. **International Journal of Scientific & Technology Research**, v. 5, n. 12. p. 140-143, 2016.

OKABE. P. E.; MASARATI. P. Modeling and simulation of a 3D printer based on a SCARA mechanism. In: LLAGUNES. J. M. **Multibody dynamics**: computational methods and applications. Barcelona: Springer, 2016. v. 37.

PAPON. A. E. HAQUE. A. SHARIF. M. A. R. Effect of nozzle geometry on melt flow simulation and structural property of thermoplastic nanocomposites in fused deposition modeling. *In:* AMERICAN SOCIETY OF COMPOSITES, 32., Lafayette, Indiana, 2017. **Anais [...]**. Lafayette: ASC, 2017.

PATEL. S.; SOBH. T. Manipulator performance measures: a comprehensive literature survey. **Journal of Intelligent and Robotic Systems**, v. 77, p. 1-27, 2015.

RASTEGAR. J.; PEREL. D. Generation of manipulator workspace boundary geometry using the Monte Carlo method and interactive computer graphics. **Journal of mechanical design**, v.112, n. 3, p. 452-454, 1990.

ROSÁRIO. J. **Princípios de mecatrônica**. São Paulo: Prentice Hall, 2005.

SCIAVICCO, L.; SICILIANO, B. **Modelling and control of robots**. 2. ed. London: Springer, 2006.

SICILIANO. B.; OUSSAMA. K. **Springer handbook of robotics**. 2. ed. Wurzburg: Springer, 2016.

SPONG. M. W.; HUTCHINSON. S.; VIDYASAGAR. M. **Robot modeling and control**. Nova Iorque: John Wiley & Sons, inc., 2006.

TAVARES. P. et al. Multi manipulator path planning using double A. **Industrial robots:** an international journal. Bingley: Emerald Group Publishing, 2016. v. 43.

VICENTE. M. F et al. Effect of ifill Parameters on tensile mechanical behavior in desktop 3D printing. **3D printing and additive manufacturing**, v. 3, n. 3, 2016.

YANG. G.; PHAM. C. B. Workspace performance optimization of fully restrained cable-driven parallel manipulator. *In*: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, n.19, 2006. **Anais [...]**. Beijing: IROS, 2006.

ZACHARIAS. F. et al. Workspace comparisons of setup configurations for human-robot interaction**.** *In*: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, n.23, 2010. **Anais [...]**. Beijing: IROS, 2010.

APÊNDICE A – CÓDIGO PRINCIPAL

```
%% Programa principal
% Recebe um contorno e retorna a divisão das tarefas e o preenchimento
clear;
clc;
close all;
dados;
 load('peca1.mat'); %cerregar peca 1
% load('peca2.mat'); %cerregar peca 1
figure(1)
plot(Outline(:,1),Outline(:,2));

%% Coordenar Outline
[Outline1 Outline2] =splitpath(Outline,ext_diam,lim_divisor,2);
% condicoes inciais
t = 0; % variavel auxiliar tempo
t_out1=0; % tempo para percorrer o contorno do manip 1
t_out2=0; % tempo para percorrer o contorno do manip 2
flipp=0; % sentido para percorrer o contorno
k=1; % variavel auxiliar
done=0; % variavel para acusar final do laco
delay=0; % delar entre o comeco do contorno 1 e 2
% tempo total acumulado
t1_acc=0;
t2_acc=0;
%tempo referencia
t1=t;
t2=t+0.1/vel2;
% iteracoes
while done==0&&k<=10
    col=0;
    t=delay;
    % testar as duas possibilidades
    for j=1:2
        if flipp==0
            temppath = Outline1;
            %Acrescer ponto inicial
            temppath = [p0_1;temppath];
            temppath2 = Outline2;
            %Acrescer ponto inicial
            temppath2 = [p0_2;temppath2];
            % retirar manip da zona de colisao
            if temppath(end,2)> lim_divisor-(geom(1,2)+geom(2,2))
                temppath(end+1,:) = [temppath(end,1) lim_divisor-
1.2*(geom(1,2)+geom(2,2))];
            end
            % calcular tempo total do manipulador 1
            if t_out1==0
                for i=2:size(Outline1,1)
                    t_out1=t_out1+norm(temppath(i,:)-temppath(i-1,:))/vel1;
                end
            end
            t=delay;
            for i=2:size(temppath2,1)
                v = temppath2(i,:)-temppath2(i-1,:);
                dv = norm(v)/1;
                v = v/norm(v);
                 for jj=1:1
                    % ponto de teste manip 2
                    p_teste = dv*jj*v+ temppath2(i-1,:);
```

```
                    t=t+dv*i/vel2;
                    tempvar =  posicao(t,temppath,vel1);
                    % verificar colisao
                    col = colisao(tempvar,p_teste,comp,base,geom,2);
                    if col==1
                        flipp=1;
                        break;
                    end
                end
                if col==1
                        break;
                end
            end
            if col==0
                % tempo acumulado
                t1_acc=t_out1;
                t2_acc=delay+t_out2;
                % salvar sequencia do contorno para cada manip
                pathout1 = temppath;
                pathout2 = temppath2;
                % salvar ponto inicial dsa proximas etapas
                path1=temppath(end,:);
                path2=temppath2(end,:);
                p2(1:2) = path2;
                p1(1:2) = path1;
                % acusar final do laco
                done=1;
                break;
            end
            t=delay;
             if flipp ==1
                temppath = [flip(Outline1(:,1)) flip(Outline1(:,2))];
                %Acrescer ponto inicial
                temppath = [p0_1;temppath];
                temppath2 = Outline2;
                %Acrescer ponto inicial
                temppath2 = [p0_2;temppath2];
                % retirar manip da zona de colisao
                if temppath(end,2)> lim_divisor-(geom(1,2)+geom(2,2))
                    temppath(end+1,:) = [temppath(end,1) lim_divisor-
(geom(1,2)+geom(2,2))];
                end
                % calcular tempo total do manipulador 1
                if t_out2==0
                    for i=2:size(Outline1,1)
                        t_out2=t_out2+norm(temppath(i,:)-temppath(i-
1,:))/vel1;
                    end
                end
                for i=2:size(Outline2,1)
                    v = temppath2(i,:)-temppath2(i-1,:);
                    dv = norm(v)/1;
                    v = v/norm(v);
                     for jj=1:1
                        % ponto de teste manip 2
                        p_teste = dv*jj*v+ temppath2(i-1,:);
                        t=t+dv*i/vel2;
                        tempvar =  posicao(t,temppath,vel1);
                        % verificar colisao
                        col = colisao(tempvar,p_teste,comp,base,geom,2);
                        if col==1
```

```
                                flipp=0;
                                % definir delay
                                delay = k*t_out1/10;
                                k=k+1;
                                break;
                            end
                        end
                        if col==1
                            break;
                        end
                    end
                    if col==0
                        % tempo acumulado
                        t1_acc=delay+t_out2;
                        t2_acc=delay+t_out2;
                        % salvar sequencia do contorno para cada manip
                        pathout1 = temppath;
                        pathout2 = temppath2;
                        % salvar ponto inicial dsa proximas etapas
                        path1=temppath(end,:);
                        path2=temppath2(end,:);
                        p2(1:2) = path2;
                        p1(1:2) = path1;
                        % acusar final do laco
                        done=1;
                        break;
                    end
                end
            end
        end
    end
end
if k>10
    % tempo acumulado
    t1_acc=t_out1;
    t2_acc=delay+t_out2;
    % salvar sequencia do contorno para cada manip
    pathout1 = temppath;
    pathout2 = temppath2;
    % salvar ponto inicial das proximas etapas
    path1=temppath(end,:);
    path2=temppath2(end,:);
    p2(1:2) = path2;
    p1(1:2) = path1;
end

%% Bounding box %%
bbox(1,1) = min(Outline(:,1));
bbox(2,1) = max(Outline(:,1));
bbox(1,2) = min(Outline(:,2));
bbox(2,2) = max(Outline(:,2));

%% Gerar infill
crop = infill(Outline,ext_diam,percent,bbox,pattern);
figure(2);
crop = sequence(crop);
hold on
plot(crop(:,1),crop(:,2));
plot(Outline(:,1),Outline(:,2),'r');

%% primeira divisao
```

```
[crop1, crop2] = splitpath(crop,ext_diam,lim_divisor,2);
[crop3, crop4] = splitpath(crop1,ext_diam,lim_divisor-
(geom(1,2)+geom(2,2)),2);
cell{1} = sequence(crop3);
cell{2} = sequence(crop4);
[crop3, crop4] =
splitpath(crop2,ext_diam,lim_divisor+(geom(1,2)+geom(2,2)),2);
cell{3} = sequence(crop3);
cell{4} = sequence(crop4);

%% gerar grafos

% colisao entre as celulas. indexado por cell.
G_colisao = [0 0 0 0;
             0 0 2 0;
             0 2 0 0;
             0 0 0 0];

 % conectividade entre as celulas do manip 1. indexado por task1
 G_conec1 = [0 1;
             1 0];

 % conectividade entre as celulas do manip 2 indexado por task2
 G_conec2 = [0 1
             1 0];

%% iteracoes

%Atribuir as celulas aos manip
task1 = [1 2];
task2 = [3 4];
%tempo referencia
t1=t1_acc;
t2=t2_acc;
%zerar condicao inicial
complete1 = 0;
complete2 = 0;
executado = zeros(size(cell,2),1);
% tarefa inicial
p1(3)=0;
p2(3)=0;
% contador da sequencia
k1 = 1;
k2 = 1;
% sequencia de tarefas
seq1(1)=0;
seq2(1)=0;
% sequencia de movimento livre
movimento1{1}=0;
movimento2{1}=0;
%backup - retorna o sistema ao momento antes de comecar a execucao da
%tarefa
bkup1{1}=t1;
bkup1{2}=p1;
bkup1{3}=executado;
bkup1{4}=seq1;
bkup1{5}=k1;
bkup1{6}=t2;
bkup1{7}=p2;
bkup1{8}=seq2;
```

```
bkup1{9}=k2;
bkup1{10}=t1_acc;
bkup1{11}=t2_acc;
bkup1{12}=movimento1;
bkup1{13}=movimento2;
bkup2{1}=t2;
bkup2{2}=p2;
bkup2{3}=executado;
bkup2{4}=seq2;
bkup2{5}=k2;
bkup2{6}=t1;
bkup2{7}=p1;
bkup2{8}=seq1;
bkup2{9}=k1;
bkup2{10}=t1_acc;
bkup2{11}=t2_acc;
bkup2{12}=movimento1;
bkup2{13}=movimento2;

%eliminar celulas vaziar
for i=1:size(cell,2)
    tempvar = cell{i};
    if tempvar(1:2,1:2) == [0 0;0 0]
        executado(i)=1;
    end
end

%% laco para coordenar as tarefas
while complete1==0||complete2==0
    split=0;
    %% mover manip 1
    if t1<=t2
        %completar task atual
        if p1(3)~=0
            executado(p1(3))=1;
        end
        %verificar se completo
        for i=1:size(task1,2)
            if executado(task1(i))==0
                break;
            end
            if i==size(task1,2)
                complete1=1;
                t2=t2-t1;
                t1=t2+1;
            end
        end
        % escolher tarefa
        if complete1==0 %verificar se ainda nao terminou suas tarefas
            % gerar malha de valores
            taskweight1 = zeros(size(task1,2),1);
            for i=1:size(task1,2)
                %verificar se ja executado
                if executado(task1(i))==1
                    taskweight1(i) = 100;
                else
                    for j=1:size(task2,2)
                        if
G_colisao(task1(i),task2(j))>=1&&executado(task2(j))==0
                            taskweight1(i) = taskweight1(i)+1;
                        end
```

```
                    end
                    %Verificar colisao
                    if p2(3)~=0
                        if G_colisao(task1(i),p2(3))>=1
                            taskweight1(i) = 100;
                        end
                    end
                end
            end
            %Escolher tarefa
            tempvar = min(taskweight1);
            %Verificar se ha tarefas possiveis
            if tempvar==100
                split=1;

            else % somente realizar se nao houver split
                %Atribuir tarefa
                tempvar = task1(find(taskweight1==tempvar,1));
                % encontrar ponto mais proximo da tarefa
                temppath = cell{tempvar};
                if norm(p1(1:2)-temppath(1,:))<=norm(p1(1:2)-
temppath(end,:))
                    tempvar2=temppath(1,:);
                    tempvar3=temppath(end,:);
                else
                    tempvar2=temppath(end,:);
                    tempvar3=temppath(1,:);
                end
                %decidir tipo de movimento baseado na adjacencia
                if p1(3) ~=0
                    if G_conec1(find(task1==p1(3)),find(task1==tempvar))>=1
% se adjacente
                        movimento1{k1}=[p1(1:2); tempvar2];
                    else
                        if p1(2)<=lim_divisor-(geom(2,2)+geom(1,2))% se nao
adjacente mas abaixo da zona de colisao
                            movimento1{k1}=[p1(1:2);
                                            [tempvar2(1) lim_divisor-
(geom(2,2)+geom(1,2))];
                                            tempvar2];
                        else
                            movimento1{k1}=[p1(1:2);
                                             [p1(1) lim_divisor-
(geom(2,2)+geom(1,2))];
                                             [tempvar2(1) lim_divisor-
(geom(2,2)+geom(1,2))];
                                             tempvar2];
                        end
                    end
                else
                    if p1(2)<=lim_divisor-(geom(2,2)+geom(1,2))
                            movimento1{k1}=[p1(1:2);
                                            [tempvar2(1) lim_divisor-
(geom(2,2)+geom(1,2))];
                                            tempvar2];
                    else
                        movimento1{k1}=[p1(1:2);
                                        [p1(1) lim_divisor-
(geom(2,2)+geom(1,2))];
                                        [tempvar2(1) lim_divisor-
(geom(2,2)+geom(1,2))];
```

```
                                        tempvar2];
                    end
                end
                % salvar novo ponto
                p1(1:2) = tempvar3;
                p1(3) = tempvar;
                %calcular tempo de movimento
                d_mov = 0;
                for i=2:size(movimento1)
                    tempvar = movimento1{i-1};
                    tempvar2 = movimento{i};
                    d_mov = d_mov + norm(tempvar2-tempvar);
                end
                %Avançar tempo
                t2 = t2-t1;
                t1 = tempo_execucao(cell{p1(3)},vel1)+d_mov/vel1;
                t1_acc = t1_acc + t1;
                %Gravar sequencia
                seq1(k1) = p1(3);
                k1=k1+1;
            end
        end
        %% mover manip 2
    else
        %completar task atual
        if p2(3)~=0
            executado(p2(3))=1;
        end
        %verificar se completo
        for i=1:size(task2,2)
            if executado(task2(i))==0
                break;
            end
            if i==size(task2,2)
                complete2=1;
                t1=t1-t2;
                t2=t1+1;
            end
        end
        % mover manipulador 2
        if complete2==0 %verificar se ainda nao terminou suas tarefas
            % gerar malha de valores
            taskweight2 = zeros(size(task2,2),1);
            for i=1:size(task2,2)
                %verificar se ja executado
                if executado(task2(i))==1
                    taskweight2(i) = -1;
                else
                    for j=1:size(task1,2)
                        if
G_colisao(task2(i),task1(j))>=1&&executado(task1(j))==0
                            taskweight2(i) = taskweight2(i)+1;
                        end
                    end
                    %Verificar colisao
                    if p1(3)~=0
                        if G_colisao(task2(i),p1(3))>=1
                            taskweight2(i) = -1;
                        end
                    end
                end
```

```
            end
            %Escolher tarefa
            tempvar = max(taskweight2);
            %Verificar se ha tarefas possiveis
            if tempvar==-1
                split=1;
            else %somente realizar se nao houver split
                %Atribuir tarefa
                tempvar = task2(find(taskweight2==tempvar,1));
                % encontrar ponto mais proximo da tarefa
                temppath = cell{tempvar};
                if norm(p2(1:2)-temppath(1,:))<=norm(p2(1:2)-
temppath(end,:))
                    tempvar2=temppath(1,:);
                    tempvar3=temppath(end,:);
                else
                    tempvar2=temppath(end,:);
                    tempvar3=temppath(1,:);
                end
                %decidir tipo de movimento baseado na adjacencia
                if p2(3) ~=0
                    if G_conec2(find(task2==p2(3)),find(task2==tempvar))>=1
                        movimento2{k2}=[p2(1:2); tempvar2];
                    else
                        if p2(2)>=lim_divisor+geom(2,2)+geom(1,2)
                            movimento2{k2}=[p2(1:2);
                                            [tempvar2(1)
lim_divisor+geom(2,2)+geom(1,2)];
                                            tempvar2];
                        else
                            movimento2{k2}=[p2(1:2);
                                             [p2(1)
lim_divisor+geom(2,2)+geom(1,2)];
                                             [tempvar2(1)
lim_divisor+geom(2,2)+geom(1,2)];
                                             tempvar2];
                        end
                    end
                else
                    if p2(2)>=lim_divisor+geom(2,2)+geom(1,2)
                            movimento2{k2}=[p2(1:2);
                                            [tempvar2(1)
lim_divisor+geom(2,2)+geom(1,2)];
                                            tempvar2];
                    else
                        movimento2{k2}=[p2(1:2);
                                        [p2(1)
lim_divisor+geom(2,2)+geom(1,2)];
                                        [tempvar2(1)
lim_divisor+geom(2,2)+geom(1,2)];
                                        tempvar2];
                    end
                end
                % salvar novo ponto
                p2(1:2) = tempvar3;
                p2(3) = tempvar;
                %calcular tempo de movimento
                d_mov = 0;
                for i=2:size(movimento2)
                    tempvar = movimento2{i-1};
                    tempvar2 = movimento{i};
```

```
                d_mov = d_mov + norm(tempvar2-tempvar);
            end
            %Avançar tempo
            t1 = t1-t2;
            t2 = tempo_execucao(cell{p2(3)},vel2)+d_mov/vel2;
            t2_acc = t2_acc + t2;
            %Gravar sequencia
            seq2(k2) = p2(3);
            k2=k2+1;
        end
    end
end
%%
if split==1
    if t1<=t2
        %escolher nao executado de 1
        tempvar = find(executado(task1)==0,size(executado,1));
        tempvar2 = min(G_colisao(task1(tempvar),p2(3)));
        temptask1 = find(G_colisao(p2(3),:)==tempvar2,1);
        temptask2 = p2(3);
    else
        tempvar = find(executado(task2)==0,size(executado,1));
        tempvar2 = min(G_colisao(task2(tempvar),p1(3)));
        temptask2 = find(G_colisao(p1(3),:)==tempvar2,1);
        temptask1 = p1(3);
    end

    temppath=cell{temptask1};
    boxpath = [min(temppath(:,1)) max(temppath(:,1));
               min(temppath(:,2)) max(temppath(:,2))];
    Area1=abs((boxpath(1,1)-boxpath(1,2))*(boxpath(2,1)-boxpath(2,2)));
    temppath=cell{temptask2};
    boxpath = [min(temppath(:,1)) max(temppath(:,1));
               min(temppath(:,2)) max(temppath(:,2))];
    Area2=abs((boxpath(1,1)-boxpath(1,2))*(boxpath(2,1)-boxpath(2,2)));
    %% split manip 1
    if Area1>=Area2
        % Split cell
        % escolher celula
        if t2<t1
            %retornar ao momento anterior
            t1=bkup1{1};
            p1=bkup1{2};
            tempvar=bkup1{3};
            executado(1:size(tempvar,1))=tempvar;
            executado(size(tempvar,1)+1:end)=0;
            seq1=bkup1{4};
            k1=bkup1{5};
            t2=bkup1{6};
            p2=bkup1{7};
            seq2=bkup1{8};
            k2=bkup1{9};
            t1_acc=bkup1{10};
            t2_acc=bkup1{11};
            movimento1=bkup1{12};
            movimento2=bkup1{13};
            executado = zeros(size(cell,2),1);
            for i=1:size(cell,2)
                tempvar = cell{i};
                if tempvar(1:2,1:2) == [0 0;0 0]
                    executado(i)=1;
```

```
                    end
                end
                if complete1==1
                    complete1=0;
                end
            end
            splittask = temptask1;
            %
            temppath = cell{splittask};
            boxpath = [min(temppath(:,1)) max(temppath(:,1));
                       min(temppath(:,2)) max(temppath(:,2))];
            % definir corte horizontal ou vertical e limiar
            if abs(boxpath(1,1)-boxpath(1,2))>=abs(boxpath(2,1)-
boxpath(2,2))
               mode = 1; %corte horizontal
               thr = (boxpath(1,2)+boxpath(1,1))/2;
            else
               mode = 2; % corte vertical
               thr = (boxpath(2,2)+boxpath(2,1))/2;
            end
            % dividir a celula
            [crop1, crop2] = splitpath(temppath,ext_diam,thr,mode);
            % atualizar cell
            cell{splittask}=sequence(crop1);
            cell{end+1}=sequence(crop2);
            % atualizar task e executados
            executado(end+1)=0;
            task1(end+1)=size(cell,2);
            %eliminar celular vaziar
            for i=1:size(cell,2)
                tempvar = cell{i};
                if tempvar(1:2,1:2) == [0 0;0 0]
                    executado(i)=1;
                end
            end
            % Atualizar grafo colisao
            G_colisao(end+1,:)=zeros(size(cell,2)-1,1);
            G_colisao(:,end+1)=zeros(size(cell,2),1);
            % verificar nova relacao entre as celulas
            for i=1:size(cell,2)
                if G_colisao(splittask,i)>=1
                   temppath = cell{i};
                   boxtemp = [min(temppath(:,mode)) max(temppath(:,mode))];
                   % zerar valor para nova avaliacao
                   G_colisao(splittask,i)=0;
                   G_colisao(i,splittask)=0;
                   % verificar se o primeiro lado esta em contato com a
                   % celula
                   temppath = cell{splittask};
                   boxpath = [min(temppath(:,mode)) max(temppath(:,mode))];
                   if boxtemp(1)>=(boxpath(1)-
(geom(1,2)+geom(2,2)))&&boxtemp(1)<=(thr+(geom(1,2)+geom(2,2)))
                       G_colisao(splittask,i)=G_colisao(splittask,i)+1;
                       G_colisao(i,splittask)=G_colisao(i,splittask)+1;
                   end
                   %verificar se o segundo lado esta em contato com a
                   %celula
                   if boxtemp(2)>=(boxpath(1)-
(geom(1,2)+geom(2,2)))&&boxtemp(2)<=(thr+(geom(1,2)+geom(2,2)))
                       G_colisao(splittask,i)=G_colisao(splittask,i)+1;
                       G_colisao(i,splittask)=G_colisao(i,splittask)+1;
```

```
                    end
                    % repetir para a outra celula gerada
                    temppath = cell{end};
                    boxpath = [min(temppath(:,mode)) max(temppath(:,mode))];
                    if boxtemp(1)>=(thr-
(geom(1,2)+geom(2,2)))&&boxtemp(1)<=(boxpath(2)+(geom(1,2)+geom(2,2)))
                        G_colisao(end,i)=G_colisao(end,i)+1;
                        G_colisao(i,end)=G_colisao(i,end)+1;
                    end
                    %
                    if boxtemp(2)>=(thr-
(geom(1,2)+geom(2,2)))&&boxtemp(2)<=(boxpath(2)+(geom(1,2)+geom(2,2)))
                        G_colisao(end,i)=G_colisao(end,i)+1;
                        G_colisao(i,end)=G_colisao(i,end)+1;
                    end
                end
            end
            % atualizar grafo conexoes
            G_conec1(end+1,:)=zeros(size(task1,2)-1,1);
            G_conec1(:,end+1)=zeros(size(task1,2),1);
            %conectar os pontos divididos
            G_conec1(end,find(task1==splittask))=1;
            G_conec1(find(task1==splittask),end)=1;
            % verificar demais pontos
            for i=1:size(task1,2)-1
                j=find(task1==splittask);
                if G_conec1(j,i)==1
                    temppath = cell{task1(i)};
                    if mode==1
                        boxtemp = [min(temppath(:,1)) max(temppath(:,1))];
                        %verificar se ha contato com a celula da direita
                        if boxtemp(2)>=thr
                            G_conec1(end,i)=1;
                            G_conec1(i,end)=1;
                            %verificar se ha contato com a celula da
                            %esqurda
                            if boxtemp(1)>thr
                                G_conec1(j,i)=0;
                                G_conec1(i,j)=0;
                            end
                        end
                    else
                        boxtemp = [min(temppath(:,2)) max(temppath(:,2))];
                        %verificar se ha contato com a celula de cima
                        if boxtemp(2)>=thr
                            G_conec1(end,i)=1;
                            G_conec1(i,end)=1;
                            %verificar se ha contato com a celula de
                            %baixo
                            if boxtemp(1)>thr
                                G_conec1(j,i)=0;
                                G_conec1(i,j)=1;
                            end
                        end
                    end
                end
            end
        %% split manip 2
        else
            %escolher celula
            if t1<=t2
```

```
                %retornar ao momento anterior
                t2=bkup2{1};
                p2=bkup2{2};
                tempvar=bkup2{3};
                executado(1:size(tempvar,1))=tempvar;
                executado(size(tempvar,1)+1:end)=0;
                seq2=bkup2{4};
                k2=bkup2{5};
                t1=bkup2{6};
                p1=bkup2{7};
                seq1=bkup2{8};
                k1=bkup2{9};
                t1_acc=bkup2{10};
                t2_acc=bkup2{11};
                movimento1=bkup2{12};
                movimento2=bkup2{13};
                executado = zeros(size(cell,2),1);
                for i=1:size(cell,2)
                    tempvar = cell{i};
                    if tempvar(1:2,1:2) == [0 0;0 0]
                        executado(i)=1;
                    end
                end
                if complete2==1
                    complete2=0;
                end
            end
            splittask = temptask2;
            temppath = cell{splittask};
            boxpath = [min(temppath(:,1)) max(temppath(:,1));
                       min(temppath(:,2)) max(temppath(:,2))];
            % definir corte horizontal ou vertical e limiar
            if abs(boxpath(1,1)-boxpath(1,2))>=abs(boxpath(2,1)-
boxpath(2,2))
               mode = 1;
               thr = (boxpath(1,2)+boxpath(1,1))/2;
            else
               mode = 2;
               thr = (boxpath(2,2)+boxpath(2,1))/2;
            end
            % dividir a celula
            [crop1, crop2] = splitpath(temppath,ext_diam,thr,mode);
            % atualizar cell
            cell{splittask}=sequence(crop1);
            cell{end+1}=sequence(crop2);
            % atualizar task e executados
            executado(end+1)=0;
            task2(end+1)=size(cell,2);
            %eliminar celular vaziar
            for i=1:size(cell,2)
                tempvar = cell{i};
                if tempvar(1:2,1:2) == [0 0;0 0]
                    executado(i)=1;
                end
            end
            % Atualizar grafo colisao
            G_colisao(end+1,:)=zeros(size(cell,2)-1,1);
            G_colisao(:,end+1)=zeros(size(cell,2),1);
            % verificar nova relacao entre as celulas
            for i=1:size(cell,2)
                if G_colisao(splittask,i)>=1
```

```
                    temppath = cell{i};
                    boxtemp = [min(temppath(:,mode)) max(temppath(:,mode))];
                    % zerar valor para nova avaliacao
                    G_colisao(splittask,i)=0;
                    G_colisao(i,splittask)=0;
                    % verificar se o primeiro lado esta em contato com a
                    % celula
                    temppath = cell{splittask};
                    boxpath = [min(temppath(:,mode)) max(temppath(:,mode))];
                    if boxtemp(1)>=(boxpath(1)-
(geom(1,2)+geom(2,2)))&&boxtemp(1)<=(thr+(geom(1,2)+geom(2,2)))
                        G_colisao(splittask,i)=G_colisao(splittask,i)+1;
                        G_colisao(i,splittask)=G_colisao(i,splittask)+1;
                    end
                    %verificar se o segundo lado esta em contato com a
                    %celula
                    if boxtemp(2)>=(boxpath(1)-
(geom(1,2)+geom(2,2)))&&boxtemp(2)<=(thr+(geom(1,2)+geom(2,2)))
                        G_colisao(splittask,i)=G_colisao(splittask,i)+1;
                        G_colisao(i,splittask)=G_colisao(i,splittask)+1;
                    end
                    % repetir para a outra celula gerada
                    temppath = cell{end};
                    boxpath = [min(temppath(:,mode)) max(temppath(:,mode))];
                    if boxtemp(1)>=(thr-
(geom(1,2)+geom(2,2)))&&boxtemp(1)<=(boxpath(2)+(geom(1,2)+geom(2,2)))
                        G_colisao(end,i)=G_colisao(end,i)+1;
                        G_colisao(i,end)=G_colisao(i,end)+1;
                    end
                    %
                    if boxtemp(2)>=(thr-
(geom(1,2)+geom(2,2)))&&boxtemp(2)<=(boxpath(2)+(geom(1,2)+geom(2,2)))
                        G_colisao(end,i)=G_colisao(end,i)+1;
                        G_colisao(i,end)=G_colisao(i,end)+1;
                    end
                end
            end
            % atualizar grafo conexoes
            G_conec2(end+1,:)=zeros(size(task2,2)-1,1);
            G_conec2(:,end+1)=zeros(size(task2,2),1);
            %conectar os pontos divididos
            G_conec2(end,find(task2==splittask))=1;
            G_conec2(find(task2==splittask),end)=1;
            %verificar demais pontos
            for i=1:size(task2,2)-1
                j=find(task2==splittask);
                if G_conec2(j,i)==1
                    temppath = cell{task2(i)};
                    if mode==1
                        boxtemp = [min(temppath(:,1)) max(temppath(:,1))];
                        %verificar se ha contato com a celula da direita
                        if boxtemp(2)>=thr
                            G_conec2(end,i)=1;
                            G_conec2(i,end)=1;
                            %verificar se ha contato com a celula da
                            %esquerda
                            if boxtemp(1)>thr
                                G_conec2(j,i)=0;
                                G_conec2(i,j)=0;
                            end
                        end
```

```
                    else
                        boxtemp = [min(temppath(:,2)) max(temppath(:,2))];
                        %verificar se ha contato com a celula de cima
                        if boxtemp(2)>=thr
                            G_conec2(end,i)=1;
                            G_conec2(i,end)=1;
                            %verificar se ha contato com a celula de
                            %baixo
                            if boxtemp(1)>thr
                                G_conec2(j,i)=0;
                                G_conec2(i,j)=0;
                            end
                        end
                    end
                end
            end
        end
    end
end

%% gerar trajetoria da etapa de infill
for i=1:size(seq1,2)
    % recuperar dados
    mov = movimento1{i};
    preenchimento = cell{seq1(i)};
    path1(end+1:end+size(mov,1),:) =  mov;
    % verificar sentido do movimento
    if mov(end,:)==preenchimento(1,:)
        path1(end+1:end+size(preenchimento,1),:) = preenchimento;
    else
        path1(end+1:end+size(preenchimento,1),:) = flip(preenchimento);
    end
end
for i=1:size(seq2,2)
    mov = movimento2{i};
    preenchimento = cell{seq2(i)};
    path2(end+1:end+size(mov,1),:) =  mov;
    % verificar sentido do movimento
    if mov(end,:)==preenchimento(1,:)
        path2(end+1:end+size(preenchimento,1),:) = preenchimento;
    else
        path2(end+1:end+size(preenchimento,1),:) = flip(preenchimento);
    end
end

%% verificar colisao na etapa de infill
col=0;
t=0;
dt=max(t1_acc-delay-t_out2,t2_acc-delay-t_out2)/div_tempo;
for i=1:1000
    t = dt*i;
    p2 = posicao(t,path2,vel2);
    p1 =  posicao(t,path1,vel1);
    col = colisao(p1,p2,comp,base,geom,2);
    if col==1
%         error('colisao no infill')
        break;
    end
end

%% verificar colisao na etapa de contorno
```

```
col=0;
t=0;
dt=max(t_out2+delay,t_out1)/div_tempo;
for i=1:1000
    t = dt*i;
    if t<delay
        p2 = pathout2(1,:);
        p1 =  posicao(t,path1,vel1);
    else
        p2 = posicao(t-delay,path2,vel2);
        p1 =  posicao(t,path1,vel1);
    end
    col = colisao(p1,p2,comp,base,geom,2);
    if col==1
        error('colisao no contorno')
        break;
    end
end

%% unificar contorno com preenchimento

path1 = [pathout1;path1];
%pathout2 = [flip(pathout2(:,1)) flip(pathout2(:,2))];
path2 = [pathout2;path2];

%% plot resultado

figure(3)
hold on;
plot(path1(:,1),path1(:,2));
plot(path2(:,1),path2(:,2),'k');

%% plot celulas
figure(4)
hold on;
for i=1:size(seq1,2)
        p = cell{seq1(i)};
        if mod(i,3)==1
            plot(p(:,1),p(:,2),'b');
        else
            if mod(i,3)==2
                plot(p(:,1),p(:,2),'r');
            else
                plot(p(:,1),p(:,2),'k');
            end
        end
end
for i=1:size(seq2,2)
        p = cell{seq2(i)};
        if mod(i,3)==1
            plot(p(:,1),p(:,2),'b');
        else
            if mod(i,3)==2
                plot(p(:,1),p(:,2),'r');
            else
                plot(p(:,1),p(:,2),'k');
            end
        end
end
```

APÊNDICE B - CÓDIGO DADOS DE ENTRADA

```
%% Dados %%

%% Manipulador 1 %% 1 = variavel da ferramenta / 2 = variavel da base
base(1,:) = [-160,-200]; % Localização da base do manipulador
comp(1,:) = [180,190]; % comprimento dos bracos do manipulador [a1,a2];
geom(1,:) = [20,25,20,25,30]; %espessura 1 / raio cabeca / espessura 2 /
raio junta 2 / raio base;
lim(1,:) = [0,100,-45,150]; %limite de angulo;

%% Manipulador 2 %% 1 = variavel da ferramenta / 2 = variavel da base
base(2,:) = [160,200]; % Localização da base do manipulador
comp(2,:) = [180,190]; % comprimento dos bracos do manipulador [a1,a2];
geom(2,:) = [20,25,20,25,30]; %espessura 1 / raio cabeca / espessura 2 /
raio junta 2 / raio base;
lim(2,:) = [80,180,-150,45]; %limite de angulo;

%% Espaco de estudo %%
s_lx = 5;
s_ly = 5;
nx = 100;
ny = 100;

lx = linspace(-s_lx,s_lx,nx);   % discretizar eixo x
ly = linspace(-s_ly,s_ly,ny);   % discretizar eixo y

[Sx Sy]=ndgrid(lx,ly);            % gerar espaço

%% Condicoes manipulador
%velocidade
vel1=1;
vel2=1;
%ponto inicial
p0_1  = [125 0];
p0_2 = [-125 0];
%% infill
ext_diam=1; % diametro do extrusor
pattern=2; % direcao do infill
percent=0.2; % densidade do infill
lim_divisor=0; % limiar de divisao da peca
%% colisao
div_tempo = 1000; % numero de divisoes do tempo para verificar colisao
```

APÊNDICE C - FUNÇÃO INFILL

```
%% infill
%% Recebe um contorno e um padrao e retorna o preenchimento

function [infill_points] = infill(Outline,ext_diam,fill_per,bbox,mode)

%% numero de divisoes

%espacamento entre retas
L = ext_diam/fill_per;
%diagonal da caixa
box_diag = sqrt((bbox(1,1)-bbox(2,1))^2+(bbox(1,2)-bbox(2,2))^2);
% numero de divisoes
n = floor(box_diag/L);

%% definicao das retas

% vetor diagonal
if mode==1
    o_diag = [bbox(2,1) bbox(1,2)];
    v_diag = [bbox(1,1) bbox(2,2)] - [bbox(2,1) bbox(1,2)]; % diag
principal
    v_diag = v_diag/norm(v_diag);
    v_infill = [1 1]/norm([1 1]); % retas a 45 graus

else
    o_diag = bbox(1,:);
    v_diag = bbox(2,:)-bbox(1,:); % segunda diag
    v_diag = v_diag/norm(v_diag);
    v_infill = [-1 1]/norm([1 1]); % retas a 135 graus

end

dl = box_diag/n;

kk=1;

for i=1:(n-1)

    o_infill(i,:)= i*dl*v_diag + o_diag;
    % zerar vetor auxiliar
    pontos_temp = [];
    k = 1;
    for j=2:size(Outline,1)
        % vetor diretor dos pontos
        v_outline = Outline(j,:) - Outline(j-1,:);
        v_outline = v_outline/norm(v_outline);
        % calcular ponto de interseccao
        determ = v_outline(1)*v_infill(2)-v_outline(2)*v_infill(1);
        numer = (o_infill(i,1)-Outline(j-1,1))*v_outline(2)-(o_infill(i,2)-
Outline(j-1,2))*v_outline(1);
        lambda = numer/determ;
        p_inter = lambda*v_infill+o_infill(i,:);
        % verificar se ha interseccao
        if norm(p_inter-Outline(j,:))<=norm(Outline(j,:) - Outline(j-
1,:))&&norm(p_inter-Outline(j-1,:))<=norm(Outline(j,:) - Outline(j-1,:))
            % guardar ponto
```

```
            pontos_temp(k,:) = p_inter;%+ext_diam*(o_infill(i,:)-
p_inter)/norm(o_infill(i,:)-p_inter);
            k = k+1;
        end
    end
    % salvar pontos
    if size(pontos_temp,1) > 0
        % ordenar pontos
        seq = sort(pontos_temp(:,1));
        if mode==2
            seq=flip(seq);
        end
        for ii=1:size(seq,1)
            j = find(pontos_temp(:,1)==seq(ii));
            pontos_temp2(ii,:)=pontos_temp(j,:);
        end
        pontos_temp = pontos_temp2;
        % dividir as descontinuidades
        for j = 1:size(pontos_temp,1)/2
            if kk>j % vetor ja existe
                v = infill_points_temp{j};
                v(end+1:end+2,:) = [pontos_temp(2*j-
1,:);pontos_temp(2*j,:)];
                infill_points_temp{j}=v;
            else % vetor nao existe
                v = [pontos_temp(2*j-1,:);pontos_temp(2*j,:)];
                infill_points_temp{j}=v;
                kk=kk+1;
            end
        end
    end
end
%% gerar output
v = infill_points_temp{1};
infill_points = v;
for i=2:size(infill_points_temp,2)
    v = infill_points_temp{i};
    infill_points(end+1:end+size(v,1),:)=v;
end
%% compensar extrusor
for i=1:size(infill_points)/2
    if mode==1
        infill_points(2*i-1,:)=infill_points(2*i-1,:)+ext_diam/2*[1
1]/norm([1 1]);
        infill_points(2*i,:)=infill_points(2*i,:)-ext_diam/2*[1 1]/norm([1
1]);
    else
        infill_points(2*i-1,:)=infill_points(2*i-1,:)+ext_diam/2*[-1
1]/norm([1 1]);
        infill_points(2*i,:)=infill_points(2*i,:)-ext_diam/2*[-1 1]/norm([1
1]);
    end
```

APÊNDICE D - FUNÇÃO CINEMÁTICA INVERSA

```
%%%% cinematica inversa %%%%

function [n,m,p] = inv_kin(p0,a,b,modo)
% n = angulo 1 (ferramenta) // m = angulo 2 (base) // p = coordenadas do
ponto do cotovelo
% p0 = ponto no workspace // a = comprimento dos bracos // b = coordandas
da base // modo = qual das duas solucoes do scara
%% 2 = angulo ferramenta // 1 = angulo base
%% "a" com indices invertidos (1=ferramenta e 2=base)
%% modo 1=teta1 positivo modo 2=teta1 negativo


x=p0(1)-b(1);
y=p0(2)-b(2);

c2=(x*x+y*y-a(1)*a(1)-a(2)*a(2))/(2*a(1)*a(2));

if c2>1
    c2=1;
end

%% verifica solucao desejada
if modo == 2
  teta2=acos(c2);
  del1=atan2(y,x);
  else
    teta2=-acos(c2);
    del1=atan2(y,x);
end

if del1<0
    del1=del1+2*pi;
end

del2=atan2(a(1)*sin(teta2),a(2)+a(1)*c2);

if del2<0
    del2=del2+2*pi;
end

teta1=del1-del2;

p(1) = a(2)*cos(teta1)+b(1);
p(2) = a(2)*sin(teta1)+b(2);

m=teta1;  %*180/pi; % angulo base
n=teta2;  %*180/pi; % angulo ferramenta
```

APÊNDICE E - FUNÇÃO POSIÇÃO DO MANIPULADOR

```
%% posicao manipulador no tempo
% recebe o tempo, uma trajetoria e a velocidade e retorna em qual ponto da
% trajetoria o manip esta
function [p] = posicao(t,seq,vel)
% t= tempo
% seq = trajetoria
% vel = velocidade

 t_real_1 = 0;
 t_real_2 = 0;
 i=2;
 % identifica entre quais dois pontos se encontra
while t_real_2<t && i<=size(seq,1)
    t_real_2 = t_real_1 + norm(seq(i,:)-seq(i-1,:))/vel;
    if t_real_2 <= t
        t_real_1 = t_real_2;
        i = i+1;
    end
end
% interpola os pontos
if i <= size(seq,1)
    dt = t - t_real_1;
    p = dt*vel*(seq(i,:)-seq(i-1,:))/norm(seq(i,:)-seq(i-1,:))+seq(i-1,:);
else
    p = seq(end,:);
end
```

APÊNDICE F - FUNÇÃO ORDENAR PONTOS DO INFILL

```
%% sequence
% ordena os pontos do infill
function [path] = sequence(path0)

n = size(path0,1)/2;
k1=1;
k2=1;

path(1,:)=path0(1,:);
path(2,:)=path0(2,:);

for i=2:n
    i1=2*i-1;
    i2=2*i;
    a = norm(path0(i1,:)-path(i1-1,:));
    b =  norm(path0(i2,:)-path(i1-1,:));
    if a < b
        path(i1,:)=path0(i1,:);
        path(i2,:)=path0(i2,:);
    else
        path(i1,:)=path0(i2,:);
        path(i2,:)=path0(i1,:);
    end
end
```

APÊNDICE G - FUNÇÃO DIVIDIR CAMINHO

```
%% split path
% recebe a trajetoria, o limiar e os dados do extrusor e divide uma
% trajetoria em duas
function [path1, path2] = splitpath(path0,ext_diam,y_thr,mode)
% path0 = trajetoria a ser cortada
% ext_diam = diametro do extrusor para correcao dos pontos
% y_thr = valor referencia de corte
% mode = modo de corte (1 = vertical, 2= horizontal)

path1 =[0 0;0 0];
path2 =[0 0;0 0];

% definir vetor de corte
if mode==2
    o_thr = [0 y_thr];
    v_thr = [1 0];
else
    o_thr = [y_thr 0];
    v_thr = [0 1];
end

n = size(path0,1)/2;
k1=1;
k2=1;

for i=1:n
    i2=2*i;
    i1=2*i-1;
    %vetor que liga dois pontos consecutivos
    v_infill = path0(i2,:)-path0(i1,:);
    v_infill = v_infill/norm(v_infill);
    % calcular ponto de interseccao
    determ = v_thr(1)*v_infill(2)-v_thr(2)*v_infill(1);
    numer = (path0(i1,1)-o_thr(1))*v_thr(2)-(path0(i1,2)-
o_thr(2))*v_thr(1);
    lambda = numer/determ;
    p_inter = lambda*v_infill+path0(i1,:);
    % verificar cada caso de posicionamento
    if path0(i1,mode) < y_thr
        if path0(i2,mode) < y_thr
            path1(k1,:)=path0(i1,:);
            path1(k1+1,:)=path0(i2,:);
            k1=k1+2;
        else
            new_point = p_inter+ext_diam/2*(path0(i1,:)-
p_inter)/norm(path0(i1,:)-p_inter);
            if norm(p_inter - path0(i1,:))> ext_diam
                path1(k1,:)=path0(i1,:);
                path1(k1+1,:)=new_point;
                k1=k1+2;
            end
            new_point = p_inter+ext_diam/2*(path0(i2,:)-
p_inter)/norm(path0(i2,:)-p_inter);
            if norm(p_inter - path0(i2,:))> ext_diam;
                path2(k2,:)=path0(i2,:);
                path2(k2+1,:)=new_point;
                k2=k2+2;
            end
        end
```

```
    else
        if path0(i2,mode) > y_thr
            path2(k2,:)=path0(i1,:);
            path2(k2+1,:)=path0(i2,:);
            k2=k2+2;
        else
            new_point = p_inter+ext_diam/2*(path0(i2,:)-
p_inter)/norm(path0(i2,:)-p_inter);
            if norm(p_inter - path0(i2,:))> ext_diam;
                path1(k1,:)=path0(i2,:);
                path1(k1+1,:)=new_point;
                k1=k1+2;
            end
            new_point = p_inter+ext_diam/2*(path0(i1,:)-
p_inter)/norm(path0(i1,:)-p_inter);
            if norm(p_inter - path0(i1,:))> ext_diam;
                path2(k2,:)=path0(i1,:);
                path2(k2+1,:)=new_point;
                k2=k2+2;
            end
        end
    end
end
```

APÊNDICE H - FUNÇÃO TEMPO DE EXECUÇÃO

```
%% tempo funcao
% recebe uma trajetoria e velocidade e retorna o tempo de execucao.

function [tempo] = tempo_execucao(path,vel)
    n = size(path,1);
    d=0;
    for i=2:n
        d = d+norm(path(i,:)-path(i-1,:));
    end
    tempo = d/vel;
```

APÊNDICE I - FUNÇÃO GERAR GEOMETRIA DO BRAÇO

```
%%%% gerar braco %%%%
% Recebe a espessura do braço e sua linha média e retorna o respectivo
% retangulo

function [m,n,o,p] = gerarbraco(teta,comp,base,geom,braco) %% p0 = ponto
ferramenta // B = ponto cotovelo // base = ponto base // geom = geometria
do manip // braco = qual o braco (1=ferramenta, 2=base)


% m -p- n     m  -B-   n
%   | |         |   |
%   | |         |   |
% p -B- o     p -base- o

B = [(comp(2)*cos(teta(2))+base(1)) (comp(2)*sin(teta(2))+base(2))];
p0 = [(comp(1)*cos(teta(1)+teta(2))+B(1))
(comp(1)*sin(teta(1)+teta(2))+B(2))];

  if braco == 1
    v_ref = p0-B; % vetor central do braco
    v_orto = [v_ref(2) -v_ref(1)]; % vetor ortogonal

    n = p0 + v_orto*geom((braco-1)*2+1)/norm(v_orto);
    m = p0 - v_orto*geom((braco-1)*2+1)/norm(v_orto);
    o = B + v_orto*geom((braco-1)*2+1)/norm(v_orto);
    p = B - v_orto*geom((braco-1)*2+1)/norm(v_orto);
  else
    v_ref = B-base;
    v_orto = [v_ref(2) -v_ref(1)];

    n = B + v_orto*geom((braco-1)*2+1)/norm(v_orto);
    m = B - v_orto*geom((braco-1)*2+1)/norm(v_orto);
    o = base + v_orto*geom((braco-1)*2+1)/norm(v_orto);
    p = base - v_orto*geom((braco-1)*2+1)/norm(v_orto);
  end
```

APÊNDICE J - FUNÇÃO AVALIAR COLISÃO

```
%%% verificar colisao %%%
% recebe um estado e as caracteristicas do sistema e retorna se ha ou nao
% colisao

function [col] = colisao(p1,p2,comp,base,geom,modo)
            % teta1, teta2, braco1, braco2

 % recuperar pontos chave
[teta1(1),teta1(2),B1]=inv_kin(p1,comp(1,:),base(1,:),modo);
[teta2(1),teta2(2),B2]=inv_kin(p2,comp(2,:),base(2,:),modo);


ponto(2,:) = B1;
ponto(1,:) = p1;
ponto(3,:) = base(1,:);
% construir braco
for i=1:2
    [braco1(i,1,:),braco1(i,2,:),braco1(i,3,:),braco1(i,4,:)] =
gerarbraco(teta1,comp(1,:),base(1,:),geom(1,:),i);
    [braco2(i,1,:),braco2(i,2,:),braco2(i,3,:),braco2(i,4,:)] =
gerarbraco(teta2,comp(2,:),base(2,:),geom(2,:),i);
end
% verificar colisao com a junta e com a ferramenta
c(1)=colisao_ponto(ponto(1,:),geom(1,2),teta2,comp(2,:),base(2,:),geom(2,:)
,braco2);
c(2)=colisao_ponto(ponto(2,:),geom(1,4),teta2,comp(2,:),base(2,:),geom(2,:)
,braco2);
c(3)=colisao_ponto(ponto(3,:),geom(1,5),teta2,comp(2,:),base(2,:),geom(2,:)
,braco2);

% verificar colisao entre os bracos
for j=1:2
  for i=1:2
    a = [braco1(j,i,:) braco1(j,5-i,:)];
    c(3+i+2*(j-1))=colisao_reta([braco1(j,i,1) braco1(j,i,2);braco1(j,5-
i,1) braco1(j,5-i,2)],teta2,comp(2,:),base(2,:),geom(2,:),braco2);
  end
end

if sum(c)>0
    col=1;
else
    col=0;

end
```

APÊNDICE K - FUNÇÃO COLISÃO PONTO-MANIPULADOR

```
%%% colisao ponto %%%
% Verifica a colisao entre manipulador e círculo
% utilizado nas bases, juntas de cotovelo e ferramenta

function [c] = colisao_ponto(p,raio,teta,comp,base0,geom,braco)

  c=0;
ponto(2,:) = [(comp(2)*cos(teta(2))+base0(1))
(comp(2)*sin(teta(2))+base0(2))];
ponto(1,1) = (comp(1)*cos(teta(1)+teta(2))+ponto(2,1));
ponto(1,2) = (comp(1)*sin(teta(1)+teta(2))+ponto(2,2));
ponto(3,:) = base0;

%% distancia ferramenta %%

dist_pferramenta=norm(ponto(1,:)-p);
if (dist_pferramenta<=(raio+geom(2)))
  c=1;
end

%% distancia cotovelo %%

dist_pcotovelo=norm(ponto(2,:)-p);
if (dist_pcotovelo<=(raio+geom(4)))
  c=1;
end

%% distancia base %%

dist_pbase=norm(ponto(3,:)-p);
if (dist_pbase<=(raio+geom(5)))
  c=1;
end

%% distancia bracos %%

for j=1:2
  for i=1:2
    dx = braco(j,i,1) - braco(j,(5-i),1);
    dy = braco(j,i,2) - braco(j,(5-i),2);
    const = braco(j,(5-i),2)*dx - braco(j,(5-i),1)*dy;
    dist_pbraco = abs(dy*p(1)+(-dx)*p(2)+const)/norm([dx -dy]);
    if dist_pbraco<=raio
      dist_crit = sqrt(comp(1)^2+geom(1)^2);
      if (norm(ponto(j,:)-p)<=dist_crit)&&(norm(ponto(j+1,:)-p)<=dist_crit)
        c=1;
      end
    end
  end
end
```

APÊNDICE L - FUNÇÃO COLISÃO RETA COM MANIPULADOR

```
%%% colisao ponto %%%
% Verifica a colisao entre manipulador e retangulos
% utilizado nos bracos do manipulador

function [c] = colisao_reta(p,teta,comp,base0,geom,braco)

c=0;
%% vetor referencia
dx = p(1,1)-p(2,1);
dy = p(1,2)-p(2,2);
v = [dx dy];
const_ref = p(2,2)*dx-p(2,1)*dy;
%%
ponto(2,:) = [(comp(2)*cos(teta(2))+base0(1))
(comp(2)*sin(teta(2))+base0(2))];
ponto(1,1) = (comp(1)*cos(teta(1)+teta(2))+ponto(2,1));
ponto(1,2) = (comp(1)*sin(teta(1)+teta(2))+ponto(2,2));
ponto(3,:) = base0;

%% distancia ferramenta %%

dist_retaferramenta=abs(v(2)*ponto(1,1)+(-
v(1))*ponto(1,2)+const_ref)/norm(v);
v2 = p(2,:)+dot(ponto(1,:)-p(2,:),(v/norm(v)))*v/norm(v);
if (norm(v2-p(2,:))<=norm(v))&&(norm(v2-p(1,:))<=norm(v))
  if (dist_retaferramenta<=(geom(2)))
    c=1;
  end
end

%% distancia cotovelo %%

dist_retacotovelo=abs(v(2)*ponto(2,1)+(-
v(1))*ponto(2,2)+const_ref)/norm(v);
v2 = p(2,:)+dot(ponto(2,:)-p(2,:),(v/norm(v)))*v/norm(v);
if (norm(v2-p(2,:))<=norm(v))&&(norm(v2-p(1,:))<=norm(v))
  if (dist_retacotovelo<=(geom(4)))
    c=1;
  end
end

%% distancia base %%

dist_retabase=abs(v(2)*ponto(3,1)+(-v(1))*ponto(3,2)+const_ref)/norm(v);

v2 = p(2,:)+dot(ponto(3,:)-p(2,:),(v/norm(v)))*v/norm(v);
if (norm(v2-p(2,:))<=norm(v))&&(norm(v2-p(1,:))<=norm(v))
  if (dist_retabase<=(geom(5)))
    c=1;
  end
end

%% distancia bracos %%

for j=1:2
  for i=1:2
    dx = braco(j,i,1) - braco(j,(5-i),1);
```

```
    dy = braco(j,i,2) - braco(j,(5-i),2);
    v2 = [dx dy];
    dx = p(2,1)-braco(j,(5-i),1);
    dy = p(2,2)-braco(j,(5-i),2);
    num = [v2;[dx dy]];
    den = [v;v2];
    lamba = det(num)/det(den);
%    const = braco(j,(5-i),2)*dx - braco(j,(5-i),1)*dy;
    p_int = lamba*v + p(2,:);
    dist_pint_A = norm(p_int-ponto(j,:));
    dist_pint_B = norm(p_int-ponto(j+1,:));
    dist_crit = sqrt(comp(1)^2+geom(1)^2);
    dist_pint_p1=norm(p(1,:)-p_int);
    dist_pint_p2=norm(p(2,:)-p_int);
%    dist_pbraco = abs(dy*p(1)+(-dx)*p(2)+const)/norm([dx -dy])
  if (dist_pint_p1<=norm(v))&&(dist_pint_p2<=norm(v))
      if (dist_pint_A<=dist_crit)&&(dist_pint_B<=dist_crit)
  %      if (norm(ponto(j,:)-p)<=dist_crit)&&(norm(ponto(j+1,:)-
p)<=dist_crit)
          c=1;
  %      end
      end
    end
  end
end
```

APÊNDICE M - FUNÇÃO PLOT BRAÇO DO MANIPULADOR

```
%%% plot braco %%%
% recebe uma configuracao do manipulador e a gera no espaço

function plotbraco(teta,comp,base0,geom,pontos) % p0 = ponto ferramenta //
cot0 = ponto cotovelo // base0 = ponto base // geom = dimensoes do manip //
pontos = pontos do braco
%%

cot0 = [(comp(2)*cos(teta(2))+base0(1)) (comp(2)*sin(teta(2))+base0(2))];
p0 = [(comp(1)*cos(teta(1)+teta(2))+cot0(1))
(comp(1)*sin(teta(1)+teta(2))+cot0(2))];


  dteta = linspace(0,2*pi,100); %gerar pontos para o circulo
  %% pontos para a cabeca (ferramenta)
  cabeca(1,:) = geom(2)*cos(dteta)+p0(1);
  cabeca(2,:) = geom(2)*sin(dteta)+p0(2);
  %% pontos para o cotovelo
  cotovelo(1,:) = geom(4)*cos(dteta)+cot0(1);
  cotovelo(2,:) = geom(4)*sin(dteta)+cot0(2);
  %% pontos para a base
  base(1,:) = geom(5)*cos(dteta)+base0(1);
  base(2,:) = geom(5)*sin(dteta)+base0(2);

  %% plot das regioes circulares
  plot(cabeca(1,:),cabeca(2,:),'k');
  hold on;
  plot(cotovelo(1,:),cotovelo(2,:),'k');
  plot(base(1,:),base(2,:),'k');
  %%
  %% plot bracos
  for i=1:2
    for j=1:1:3
      line([pontos(i,j,1) pontos(i,1+j,1)],[pontos(i,j,2)
pontos(i,1+j,2)]);
    end
      line([pontos(i,1,1) pontos(i,4,1)],[pontos(i,1,2) pontos(i,4,2)]);
  end
```