FORMALISMO PARA RESOLUÇÃO DE TAREFAS EM ROBÓTICA UTILIZANDO OTIMIZAÇÃO

Email:

Abstract— In this paper we present a formulation to use optimization theory in order to perform robot movement control. It is shown how the objective function of the optimization problem is designed in order to control robots, and how the constraints of optimization problem are used to model the physical limitations of the enviroment and of the robot itself. Moreover, we simulated the strategy, showing its technical feasibility.

Keywords— Robotics, Control, Optimization

Resumo— Este artigo descreve uma formulação para usar teoria de otimização de forma a fazer controle de movimentos com robôs. É mostrado como a função objetivo do problema de otimização é descrita de forma a ser capaz de controlar robôs, e como as restrições do problema de otimização podem ser utilizadas para modelar as limitações físicas do ambiente e do robô. Além disso, simulamos e validamos essa abordagem, mostrando sua viabilidade técnica.

Palavras-chave— Robótica, Controle, Otimização

1 Introdução

Controle de robôs é uma área de bastante foco atualmente. Diversas estratégias têm sido propostas de forma a obter sistemas robóticos cada vez mais autônomos, capazes de realizar tarefas mais complexas. Um dos problemas de controle que apresenta grandes aplicações em robótica é quando queremos que o robô alcance uma determinada configuração, uma *tarefa*, enquanto respeita algumas restrições, como por exemplo, limite de velocidade nas configurações, não-holonomias, evitamento de obstáculos, entre outras. Normalmente, a configuração desejada não é colocada de maneira explícita, mas sim como uma solução de uma equação envolvendo uma função do espaço de configurações - a *função de tarefa* - que por sua vez é normalmente obtida utilizando a cinemática direta do robô.

Uma abordagem clássica para resolver esse tipo de problema é dividida em três etapas hierarquicamente organizadas tal como mostra a hierarquia da esquerda da Figura (1). A etapa de *cinemática inversa* consiste em encontrar a configuração que atenda a tarefa especificada. A etapa de *planejamento de trajetória* consiste em calcular a trajetória a ser percorrida pelo robô para sair da configuração inicial do robô até a configuração final definida pela cinemática inversa, observando as restrições na configuração e na velocidade da configuração. Já a etapa de *controle* consiste fazer o controle em malha fechada do robô para atender a trajetória especificada na etapa anterior. Essa abordagem tem algumas desvantagens, principalmente nas duas primeiras etapas da hierarquia. A cinemática inversa pode não ser simples de ser resolvida, o que é bastante comum para robôs mais complexos tal como humanoides, além do custo computacional do planejamento de trajetórias poder ser elevado. Outra desvantagem é que a abordagem clássica não apresenta um comportamento reativo, ou seja, o robô não reage às eventuais mudanças do ambiente que não foram consideradas durante o planejamento.

Objetivando mitigar esses problemas, muitos trabalhos recentemente (Kanoun et al., 2011; Escande et al., 2014; Gonçalves et al., 2016; Bodily et al., 2017; Li et al., 2017; Quiroz-Omaña and Adorno, 2017) utilizam uma estrategia *reativa* que integra as três etapas da abordagem clássica (hierarquia da direita da Figura (1)). Essa abordagem procura resolver a tarefa utilizando um problema de otimização local, ou seja, usando informações locais da função de tarefa e dos obstáculos, levando em consideração as restrições do robô (planejamento de trajetória) enquanto objetiva realizar a tarefa, ou seja, alcançar uma configuração que resolva a equação envolvendo a função de tarefa (cinemática inversa). O resultado desse problema de otimização é uma ação de controle que pode ser executada diretamente no robô (controle).

Essa abordagem tem uma série de vantagens, entre as quais destacamos a possibilidade de modelar as limitações do ambiente e do robô de forma simples, inserindo-as como restrições no problema de otimização; seu baixo custo computacional de execução; e sua capacidade de ser reativa, ou seja, caso o ambiente sofra modificações durante a movimentação do robô, como por exemplo, a inserção de um novo obstáculo no percurso do robô, é possível considerar esse novo obstáculo, adicionando uma nova restrição no problema de otimização em tempo real. No entanto, essa abordagem tem também uma desvantagem: por ser uma estratégia local, o problema de otimização pode levar o robô para uma configuração que não é desejada - *um mínimo local* - em que o robô é ordenado a parar, porém a tarefa ainda não foi completada.

Neste trabalho, focaremos no controle *cinemático* de robôs - isto é, estaremos desconsiderando os parâmetros inerciais - objetivando reali-

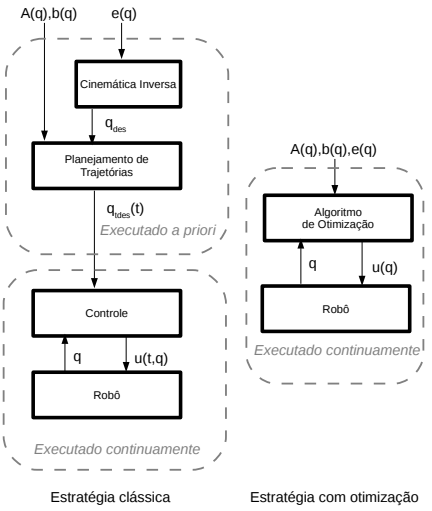

Figura 1: Abordagem clássica x abordagem com otimização. Os parâmetros $A(q), b(q)$ codificam as restrições na atuação u, como $A(q)u \leq b(q)$, e $e(q) = 0$ representa a tarefa. A função $e(q)$ é essencialmente obtida por uma cinemática direta. O vetor q_{des} é uma solução de $e(q) = 0$ e $q_{des}(t)$ é uma trajetória que leva da configuração inicial à q_{des} respeitando as restrições de atuação.

zar tarefas. O controle cinemático é aplicável em situações onde o robô é suficientemente rígido e não há forças muito elevadas atuando nele. Mais especificamente, focaremos no controle de cinemático de primeira ordem, em que a ação de controle é diretamente a velocidade da configuração. Apresentamos um arcabouço para resolução de tarefas, baseado em otimização, utilizando programação quadrática. Apresentaremos também provas formais de sua estabilidade no sentido de Lyapunov e sua aplicação em um ambiente de simulação.

2 Formalismo para controle cinemático

2.1 Ideia geral

Seja $\mathcal{Q}$ o espaço de configurações do robô, que assumimos por simplificação ser um espaço Euclidiano, isso é, $\mathcal{Q} = \mathbb{R}^n$. Seja $q \in \mathcal{Q}$ a configuração do robô e $u \in \mathbb{R}^n$ a velocidade de configuração. Assumimos um modelo cinemático de primeira ordem

$$\dot{q} = u \tag{1}$$

e também restrições para as ações de controle, as velocidades, que serão escritas utilizando desigualdades lineares em u

$$A(q)u \leq b(q) \tag{2}$$

para funções $A : \mathcal{Q} \mapsto \mathbb{R}^{r \times n}$, $b : \mathcal{Q} \mapsto \mathbb{R}^r$. Essas restrições podem modelar evitamento de colisão com obstáculos, limite na velocidade da configuração, limite das configurações, restrições de equilíbrio do robô, não-holonomias, entre outras. Será visto mais a frente como podemos modelar restrições que são somente em q, como evitamento de obstáculos e limite de juntas, no formalismo da Equação (2).

Nosso objetivo é alcançar um conjunto de configurações desejadas, $\mathcal{Q}_{des}$, uma *tarefa*, que pode ser descrita implicitamente como zeros de uma chamada *função de tarefa* (Samson et al., 1991). Essa função $e : \mathcal{Q} \mapsto \mathbb{R}^s$ descreve quais configurações são desejadas e normalmente é computada utilizando cinemática direta. Por exemplo, seja $p(q)$ a posição do efetuador de um robô manipulador, obtido pela cinemática direta, e p_d a posição no espaço de trabalho desejada, podemos então definir $e(q) = p(q) - p_d$, como a função de tarefa para esse sistema. Formalmente:

$$\mathcal{Q}_{des} = \{q \in \mathcal{Q} \mid e(q) = 0\}. \tag{3}$$

Para resolver esse problema, isso é, encontrar um u que respeite as restrições na Equação (2) e que guie a configuração para $\mathcal{Q}_{des}$, usaremos uma estrategia inspirada naquela presente em (Gonçalves et al., 2016). Nessa estrategia, tentamos forçar a dinâmica $\dot{e}(q) = -\Lambda e(q)$ para uma matriz positiva definida Λ, o que garante um decaimento exponencial assintótico da função de tarefa para 0. Essa equação pode ser reescrita, utilizando a regra da cadeia e a dinâmica na Equação (1), como

$$\dot{e}(q) + \Lambda e(q) = \frac{\partial e}{\partial q}(q)u + \Lambda e(q) = 0. \tag{4}$$

Infelizmente, nem sempre há uma solução para equação anterior que respeite as restrições na Equação (2). Visando resolver esse problema, tentaremos resolver o máximo possível da equação (4) enquanto respeitamos as restrições em (2). Portanto, formulamos o seguinte *problema de otimização.*

$$\begin{aligned} \min_{u} \quad & \left\| \frac{\partial e}{\partial q}(q)u + \Lambda e(q) \right\|^2 \\ \text{sujeito a} \quad & A(q)u \leq b(q) \end{aligned} \tag{5}$$

em que a norma $\|\cdot\|$ é a norma Euclidiana. Note que quando a função objetivo é zero a equação (4) é completamente satisfeita.

Em se tratando de sistemas redundantes, ou seja, quando os graus de liberdade da tarefa a ser executada é menor que os graus de liberdade do robô, existe a possibilidade do problema de otimização em (5) ter mais de uma solução. Assim, podemos agregar funções objetivo secundarias ao problema anterior. Uma função objetivo secundária razoável é exigir a menor norma Euclidiana possível de u. Uma maneira de implementar essa função objetivo secundária é adicionar na função objetivo de (5) um termo $\rho\|u\|^2$ com $\rho > 0$ muito

pequeno. Então, um novo problema de otimização pode ser formulado:

$$\begin{aligned} \min_{u} \quad & \left\| \frac{\partial e}{\partial q}(q)u + \Lambda e(q) \right\|^2 + \rho\|u\|^2 \\ \text{sujeito a} \quad & A(q)u \leq b(q). \end{aligned} \tag{6}$$

Como ρ é um numero muito pequeno, é mais vantajoso para o problema de otimização minimizar a função objetivo principal o máximo possível. Quando a mesma não puder mais ser melhorada, então o problema de otimização tentará atender a função objetivo secundária *perturbando muito pouco* a função objetivo primária. Quando $\rho \to 0$ só melhoraremos a função objetivo secundária se isso não causar nenhum detrimento à função objetivo primária.

O problema de otimização em (6) é um problema de programação quadrática *estritamente* convexo pois o termo $\rho\|u\|^2$ é estritamente convexo. Portanto, assumindo que o espaço factível $A(q)u \leq b(q)$ não é vazio, existe apenas *um* $u = u^*(q)$ que é solução para o problema. Temos portanto, a equação diferencial em malha fechada $\dot{q} = u^*(q)$. Podemos mostrar que esse sistema em malha fechada é Lyapunov estável.

Proposição 1 *Se o conjunto factível $\mathcal{U}$ sempre conter a solução $u = 0$, ou seja, $b(q) \geq 0$, temos que o sistema $\dot{q} = u^*(q)$ é Lyapunov estável.*

Prova: A prova é baseada naquela apresentada em (Gonçalves et al., 2016).

Seja uma função candidata de Lyapunov $V(q) = \frac{1}{2}e(q)^T\Lambda e(q)$. Ela é igual a zero se e somente se $e(q) = 0$, porque $\Lambda > 0$, enquanto se $e(q) \neq 0$ a função é maior que zero. A derivada temporal dessa função candidata é $\dot{V}(q) = e(q)^T\Lambda\frac{\partial e}{\partial q}(q)u^*(q)$. Para ser Lyapunov estável, mostraremos que $\dot{V}(q) \leq 0$.

Seja o problema de otimização em (6). Como o espaço factível tem o elemento $u = 0$, então podemos afirmar que a solução ótima u^* sempre será tal que:

$$\begin{aligned} &\left\| \frac{\partial e}{\partial q}u^* + \Lambda e \right\|^2 + \rho\|u^*\|^2 \leq \\ &\left\| \frac{\partial e}{\partial q}0 + \Lambda e \right\|^2 + \rho\|0\|^2 = \|\Lambda e\|^2. \end{aligned} \tag{7}$$

Mas notamos que :

$$\left\| \frac{\partial e}{\partial q}u^* + \Lambda e \right\|^2 = \left\| \frac{\partial e}{\partial q}u^* \right\|^2 + 2e^T\Lambda\frac{\partial e}{\partial q}u^* + \|\Lambda e\|^2. \tag{8}$$

Então, usando essa informação em (7) e simplificando

$$\left\| \frac{\partial e}{\partial q}u^* \right\|^2 + \rho\|u^*\|^2 + 2e^T\Lambda\frac{\partial e}{\partial q}u^* \leq 0. \tag{9}$$

Como os dois primeiros termos são obviamente não negativos, concluímos que:

$$e^T\Lambda\frac{\partial e}{\partial q}u^* = \dot{V}(q) \leq 0. \tag{10}$$

E a prova está completa. □

2.2 Restrições

As matrizes A e b modelam as restrições da tarefa. Essas restrições podem ser, essencialmente, de três naturezas diferentes:

- restrição na ação de controle u;
- restrição no espaço de configurações $\mathcal{Q}$;
- restrições Pfaffianas.

A restrição na ação de controle, que no caso é a velocidade $\dot{q}$, é facilmente descrita como $-w \leq \dot{q} \leq w$, em que w é um vetor positivo, supondo que os limites superiores e inferiores de $\dot{q}$ sejam iguais e simétricos. Para transformar a restrição acima no formato da equação (6) temos as duas inequações $-\dot{q} \leq w$ e $\dot{q} \leq w$. Ressalta-se que $\dot{q} = 0$ é uma solução factível, ou seja, é permitido o robô parar.

Podemos escrever restrições no espaço de configurações (como limite de juntas, evitamento de obstáculos, equilíbrio do robô, etc...) utilizando uma *função de restrição* $G : \mathcal{Q} \mapsto \mathbb{R}^l$ como $G(q) \leq 0$. Entretanto, para incluirmos esse tipo de restrição no formalismo proposto, precisamos escrevela como na Equação (2). Para isso, utilizaremos a técnica proposta em (Kanoun et al., 2011), que consiste em colocar a imposição

$$\dot{G}(q) \leq -\eta G(q) \tag{11}$$

que pode ser escrita como $\frac{\partial G}{\partial q}(q)u \leq -\eta G(q)$, ou seja, como na Equação (2). Note que essa imposição garante que, se $G(q(0)) \leq 0$ (se a configuração inicial respeita a restrição), então $G(q(t)) \leq 0$ para todo $t > 0$.

As restrições Pfaffianas naturalmente já são escritas no formato $C(q)\dot{q} = 0$. Para transformálas em desigualdades, basta fazer $C(q)\dot{q} \leq 0$ e $-C(q)\dot{q} \leq 0$. A integração das restrições de não-holonomia Pfaffianas como restrições em um problema de otimização aparece em (Quiroz-Omaña and Adorno, 2016).

3 Aplicações

3.1 O robô

Para validar essa abordagem, foi utilizado um simulador visual de um robô manipulador sob uma base móvel chamado de LittleJohn, disponível no laboratório MACRO da UFMG. Esse robô tem um manipulador com cinco juntas rotativas, $\theta_1, \ldots, \theta_5$ e uma base móvel não-holonômica

com configuração x, y, ϕ, as coordenadas do centro da base e a orientação do robô com relação à um *frame* fixo no mundo. Portanto nesse caso $q = [x\ y\ \phi\ \theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5]^T$, e o espaço de configurações $\mathcal{Q}$ e o espaço de atuação $\mathcal{U}$ são espaços $\mathbb{R}^8$. Por sua praticidade em posteriormente utilizar essa estratégia de controle no robô real, o código foi escrito em C++ como um nó do sistema ROS (Quigley et al., 2009). Esse *framework* disponibiliza uma aplicação de visualização de robôs chamada *RViz*. Utilizando um conjunto de nós, foi possível simular o robô como um integrador simples nessa aplicação.

Figura 2: Robô LittleJohn - (`http://macro.ppgee.ufmg.br/site-map/articles/14-front-end-articles/121-little-john`)

A biblioteca de otimização utilizada foi a CPLEX studio[1], que disponibiliza funções para os mais diferentes tipos de problemas de otimização, programação linear (LP), programação quadrática (QP), programação inteira-mista (MIP), etc.

3.2 A função de tarefa

Foi proposto uma função de tarefa de controle de posição com orientação (pose). A função de tarefa é dividida em duas parcelas: uma parcela da tarefa de posição e uma parcela da tarefa de orientação. A parcela da tarefa de posição é o erro entre a posição do efetuador no espaço $p(q)$ e a posição desejada no espaço de trabalho p_d. Seja $\|\cdot\|_F$ a norma de Frobenius de uma matriz (raiz da soma dos quadrados de todos os elementos). A parcela da tarefa de orientação é definida como $\left\|I - R(q)R_d^{-1}\right\|_F$, em que R e R_d são respectivamente as matrizes de rotação do efetuador e da tarefa, e I é a matriz identidade de ordem 3. Note que esse termo é 0 se e somente se $R(q) = R_d$.

Então:

$$e(q) = \begin{bmatrix} p(q) - p_d \\ \|I - R(q)R_d^{-1}\|_F \end{bmatrix}. \qquad (12)$$

[1] https://www.ibm.com/br-pt/marketplace/ibm-ilog-cplex

A posição e a orientação do efetuador foram obtidas pelo modelo cinemático direto, através dos parâmetros de Denavit-Hartenberg do robô.

3.3 As restrições

A primeira restrição que pode ser descrita na forma $G(q) \leq 0$ é a restrição de evitamento de obstáculos. Seja um função $F : \mathcal{Q} \mapsto \mathbb{R}^k$ que calcula a distância do robô aos obstáculos do ambiente, e que tenha a seguinte característica:

$$F(q) > 0 \ \Rightarrow \ \textit{robô fora do obstáculo}$$
$$F(q) \leq 0 \ \Rightarrow \ \textit{robô dentro do obstáculo}$$

Então, é suficiente considerarmos $G(q) = -F(q) + F_{mar}$, em que F_{mar} é uma margem de segurança. Neste trabalho usaremos $F_{mar} = 0,05m$.

Uma das formas de calcular a distância entre o robô e os obstáculos é: cobrir o robô e os obstáculos com esferas, e calcular a distância relativa entre a esfera do robô e as esferas dos obstáculos. Essa abordagem tende a ser muito conservadora, porque a esfera precisa ter tamanho suficiente para proteger os obstáculos e o robô totalmente. No caso do robô, uma melhoria pode ser feita: como o LittleJohn é um robô manipulador sob uma base móvel, é possível cobrir a base móvel com uma esfera, e cada elo do manipulador com outras esferas. Isso diminui um pouco o conservadorismo, na hora de desviar dos obstáculos. Neste trabalho foram consideradas seis esferas: uma pra base móvel, quatro para os elos do manipulador e uma para o efetuador. Note que o centro delas dependem da configuração do mesmo. Já o raio é fixo, e no caso foi calculado de acordo com os parâmetros dimensionais do LittleJohn. Foi considerado somente um obstáculo no espaço de trabalho. A distância relativa de cada esfera i, $i = 1, 2, ..., 6$, é dada por:

$$F_i(q) = ||C_i(q) - C_{obs}|| - (R_i + R_{obs})$$

em que $C_i(q)$ e R_i são, respectivamente, o centro (que depende da configuração e pode ser obtido com cinemática direta) e o raio da i-ésima esfera de cada parte do robô, enquanto C_{obs} e R_{obs} são, respectivamente, os centros e raios da esfera obstáculo. No nosso exemplo $R_{obs} = 0,5m$ e $C_{obs} = [2\ \ 3,5\ \ 0,5]^T m$.

A segunda restrição que pode ser descrita pela expressão $G(q) \leq 0$ é o limite nas configurações. Ela pode ser colocada como $-K \leq q \leq K$, em que K é um vetor positivo, supondo que os limites superiores e inferiores de q sejam iguais e simétricos. Para colocar no formato $G(q) \leq 0$ separamos a inequação anterior em duas inequações e fazemos: $-q - K \leq 0$ e $q - K \leq 0$.

A limitação nas configurações permite evitar a auto-colisão do manipulador do robô, além de, no

caso de bases móveis, delimitar o espaço fechado disponível para ele se movimentar. Os limites para o LittleJohn são apresentadas na Tabela 1:

Tabela 1: Limites das configurações

Configuração	**limite**
x	$[-5,0\ 5,0]\ m$
y	$[-7,0\ 7,0]\ m$
ϕ	Não tem
$\theta_1, ..., \theta_5$	$[-\pi/2\ \pi/2]\ rad$

As restrições diretamente em $\dot{q}$, as velocidades limites das configurações, foram divididas em duas partes: uma para posição da base e outra para o restante das configurações, tal como a Tabela 2.

Tabela 2: Limites de velocidade das configurações

Ação de controle	**limite**
$\dot{x}$ e $\dot{y}$	$[-0,2\ 0,2]\ m/s$
Demais	$[-1,0\ 1,0]\ rad/s$

A restrição Pffafiana é definida como $C(q)\dot{q} = 0$, onde $C(q)$ é uma matriz $C : \mathcal{Q} \mapsto \mathbb{R}^{b\times n}$. No caso em estudo, $b = 1$ porque existe somente uma restrição Pffafiana, a de não holonomia da base móvel do robô. A restrição não holonômica de robôs móveis é escrita como $sin(\phi)\dot{x} - cos(\phi)\dot{y} = 0$, então a matriz $C(q)$ é:

$$C(q) = [sin(\phi)\ -cos(\phi)\ 0\ 0\ 0\ 0\ 0\ 0].$$

Convém comentar que as restrições são tais que $u = 0$ é sempre factível se estivermos fora dos obstáculos e respeitando os limites de configuração. Se começarmos de modo que essas restrições serão satisfeitas, o que é absolutamente razoável, elas sempre serão satisfeitas para todo $t \geq 0$. Isso garante que $u = 0$ sempre é factível, a condição necessária para a aplicação da Proposição 1, que garante a estabilidade de Lyapunov.

4 Resultados Experimentais

Foi feito uma simulação da estratégia proposta usando o parâmetro $\Lambda = 3,0I$, em que I é a matriz identidade de ordem 4. A simulação visa observar o comportamento do robô no caso de não ter obstáculos no seu percurso até a tarefa desejada, e o caso em que existem obstáculos no percurso.

As tarefas estão descritas na Tabela 3. A escolha das tarefas permite avaliar o comportamento da estratégia para o caso em que a tarefa é livre de obstáculos, como é o caso das duas primeiras tarefas. Também permite avaliar o comportamento da estratégia para o caso em que a existe um obstáculo no meio do percurso do robô até a tarefa, como é o caso das duas ultimas tarefas.

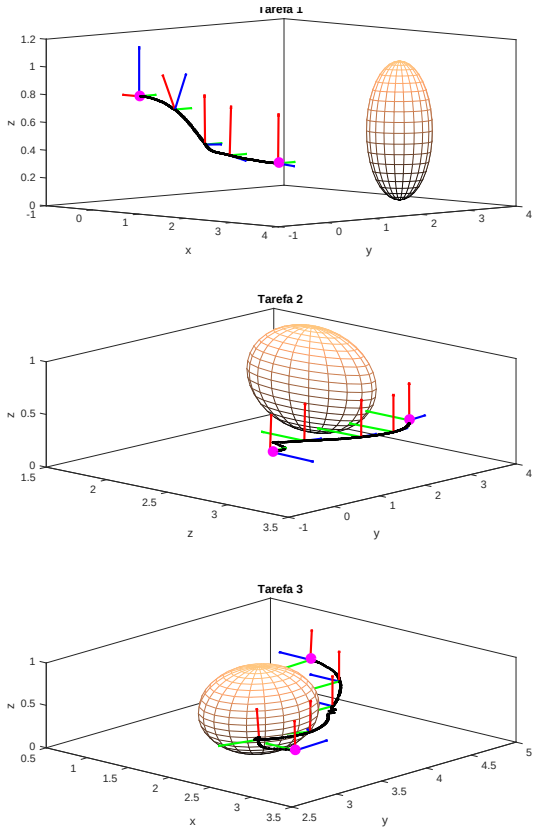

Figura 3: Percurso da pose do efetuador ao longo das três primeiras tarefas. Em preto temos o movimento da posição do efetuador. O eixo x está em azul, y em verde e z em vermelho.

Tabela 3: Parâmetros para a função de tarefa na Equação (12). $p_d = [x_d\ y_d\ z_d]^T$ e R_d é uma rotação no eixo z de um ângulo ψ

Tarefa	$x_d(m)$	$y_d(m)$	$z_d(m)$	$\psi(rad)$
1	3	0	0,4	0
2	3	3	0,4	$\pi/2$
3	1	5	0,4	π
4	2,5	3	0,4	$3\pi/2$

A disposição das três primeiras tarefas também é mostrado na Figura 3. Nessa figura, também foi colocado o movimento da pose do efetuador ao longo da trajetória obtido com a estratégia proposta.

A Figura 4 mostra o comportamento da função de Lyapunov durante as quatro tarefas mencionadas. Como é possível ver na figura, nos três primeiros casos a função converge para zero, caracterizando a realização da tarefa totalmente, porém no quarto caso, ele converge para um número maior que zero. Isso acontece quando o algoritmo de otimização chega a um mínimo local, ou seja, ele não consegue mais reduzir a função de custo sem violar as restrições. Nesse caso aparentemente o motivo do mínimo local é devido ao obstáculo na frente do robô, como é visto na figura (3).

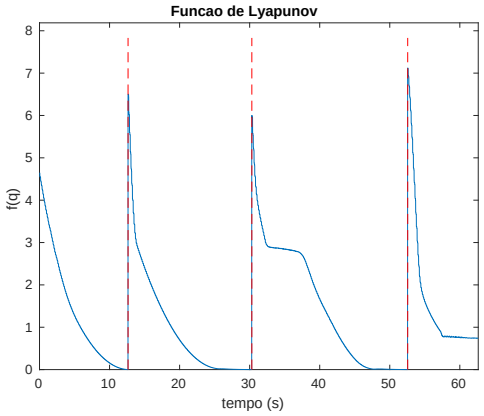

Figura 4: Função de Lyapunov $V(q) = e(q)^T \Lambda e(q)$ do sistema

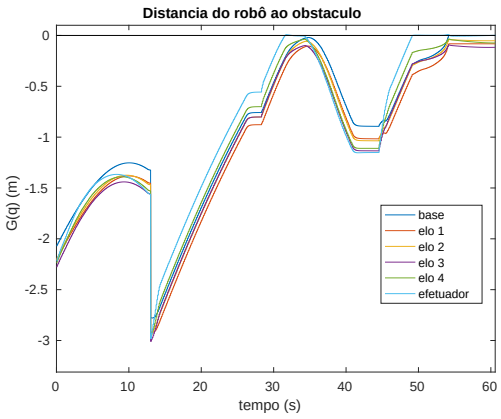

Figura 5: Distância do obstáculo - Função $G(q)$

Avaliando se o sistema está respeitando as restrições, vemos que a restrição de não holonomia é respeitada, pois analisando o comportamento das ações de controle e do ângulo ϕ da base móvel após o instante 12,63 segundos (ver Tabela 4): o robô completou a primeira tarefa, então ele tem que se deslocar totalmente para a esquerda para realizar a segunda tarefa. Portanto, era de se esperar que na ação de controle houvesse velocidade em $\dot{y}$, o que não ocorre, porque como o ângulo ϕ da base móvel ainda é igual a zero, a restrição não holonômica impede o robô ter velocidades em $\dot{y}$. Dessa forma, é necessário primeiro ter velocidade angular $\dot{\phi}$ para mudar o valor de ϕ, para então ter velocidade em $\dot{y}$, o que de fato acontece.

Tabela 4: Não Holonomia

Instante	ϕ	$\dot{y}$	$\dot{\phi}$
12,63	0	0	0
12,64	0	0	1
12,74	0.11	0,02	1

A restrição de evitamento de obstáculos é verificada pela Figura 3, que mostra que o robô desvia do obstáculo para alcançar a terceira tarefa. Outra prova é a Figura 5, que mostra o comportamento da função $G(q)$, que como é visto na figura, é sempre menor ou igual a zero.

Para verificar se o sistema está respeitando os limites de velocidades das configurações e os limites das configurações, as Figuras (6), (7), (8), (9) mostram o comportamento das velocidades e das configurações durante a realização das tarefas. Em todas as figuras é possível ver que os limites foram respeitados.

O tempo médio gasto em cada iteração do algoritmo foi 11 milissegundos, mostrando que a abordagem tem realmente um baixo custo computacional. A simulação foi feita em um computador Intel i7 2,4GHz, 16GB de memória RAM.

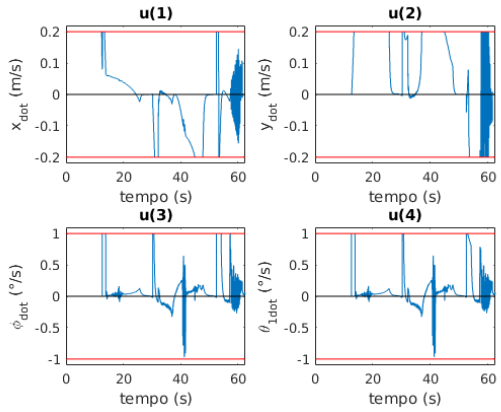

Figura 6: Velocidade das configurações - $[\dot{x}\ \dot{y}\ \dot{\phi}\ \dot{\theta_1}]$

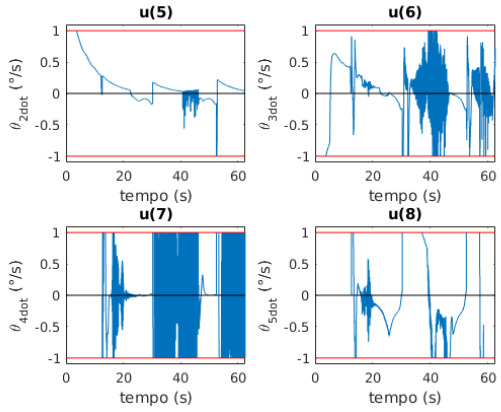

Figura 7: Velocidade das configurações - $[\dot{\theta_2}\ \dot{\theta_3}\ \dot{\theta_4}\ \dot{\theta_5}]$

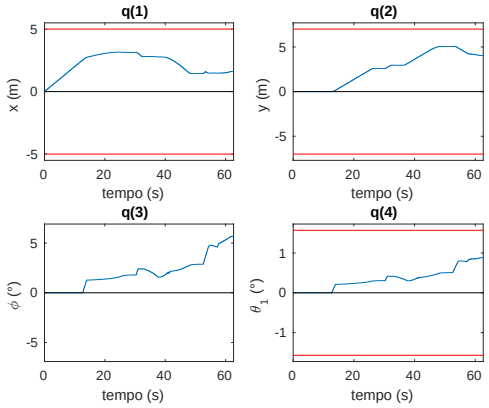

Figura 8: Configurações - $[x\ y\ \phi\ \theta_1]$

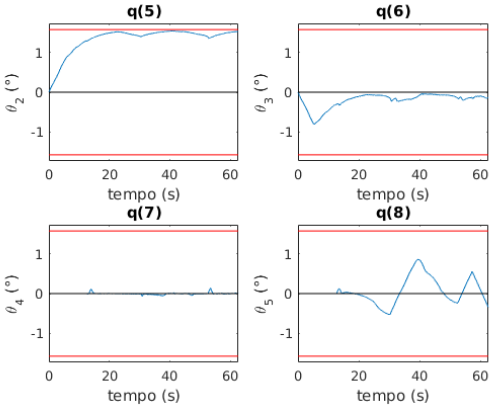

Figura 9: Configurações - $[\theta_2\ \theta_3\ \theta_4\ \theta_5]$

5 Conclusões

Neste artigo, apresentamos um formalismo para controle de movimento de robôs utilizando teoria de otimização, onde mostramos como descrever a função objetivo do problema de otimização e sua prova de estabilidade. Apresentamos também como descrever as limitações físicas do ambiente e do robô como restrições no problema de otimização. Além disso, foi mostrada uma simulação dessa abordagem, e como o controle conseguiu completar a maioria das tarefas, respeitando todas as restrições e com um baixo custo computacional. No entanto, vimos que nem sempre o robô consegue completar a tarefa, caracterizando um mínimo local. Uma das soluções para isso seria integrar, juntamente com essa estratégia, um planejador local probabilístico (LaValle, 2006), de forma que robô possa sair desse mínimo local. Deixamos isso como uma proposta de trabalho futuro.

Agradecimentos

Agradecemos à FAPEMIG e o CNPQ pela ajuda financeira concedida ao presente trabalho.

Referências

Bodily, D. M., Allen, T. F. and Killpack, M. D. (2017). Motion planning for mobile robots using inverse kinematics branching, *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, pp. 5043–5050.

Escande, A., Mansard, N. and Wieber, P.-B. (2014). Hierarchical quadratic programming: Fast online humanoid-robot motion generation, *The International Journal of Robotics Research* **33**(7): 1006–1028.

Gonçalves, V. M., Fraisse, P., Crosnier, A. and Adorno, B. V. (2016). Parsimonious kinematic control of highly redundant robots, *IEEE Robotics and Automation Letters* **1**(1): 65–72.

Kanoun, O., Lamiraux, F. and Wieber, P. B. (2011). Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task, *IEEE Transactions on Robotics* **27**(4): 785–792.

LaValle, S. M. (2006). *Planning Algorithms*, Cambridge University Press, New York, NY, USA.

Li, S., Zhang, Y. and Jin, L. (2017). Kinematic control of redundant manipulators using neural networks, *IEEE transactions on neural networks and learning systems* **28**(10): 2243–2254.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. Y. (2009). Ros: an open-source robot operating system, *ICRA Workshop on Open Source Software*.

Quiroz-Omaña, J. J. and Adorno, B. V. (2016). Parsimonious kinematic control of nonholonomic mobile manipulators, *Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2016 XIII Latin American*, IEEE, pp. 73–78.

Quiroz-Omaña, J. J. and Adorno, B. V. (2017). Whole-body kinematic control of nonholonomic mobile manipulators using linear programming, *Journal of Intelligent & Robotic Systems* .

Samson, C., Espiau, B. and Borgne, M. L. (1991). *Robot Control: The Task Function Approach*, Oxford University Press.