UNIVERSITÉ CLERMONT AUVERGNE

École Universitaire de Physique et d'Ingénierie

AprilTag-based Trajectory Tracking for the LIMOS Robot

Second-year Master's project

Automation, Robotics track: Artificial Perception and Robotics

Authors:
Joao Pedro MARTINS DO LAGO REIS
Yann Kelvem DA SILVA RAMOS

Supervisor:
Dr. Sébastien LENGAGNE

Defense Date:
March 02, 2026

Aubière, Clermont-Ferrand

Title

Subtitle

Author

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Contents

1 Introduction

Being able to follow a path once and repeat it optimally is a practical skill for mobile robots, especially in indoor environments where GPS is not available and tasks are repetitive by nature. Thus, this is the central idea behind Teach and Replay[1], the robot first performs a guided route while it registers the markers looking for a positioning reference of each tag and then uses this record of positions to reproduce the same route optimally.

In this practical exercise, this concept was implemented on a LIMO mobile robot using its integrated RGB camera and AprilTag [2] fiducial markers positioned along a closed circuit with a start and end. During the teach phase, the robot completes a first turn while detecting the markers and saves the information from the positions of the robot in relation to the tags to represent the executed movement. After pressing the Y key of the Xbox controller or another controller's triangle, the system switches to the replay phase, where the robot tries to follow the learned trajectory. The goal is simple: complete the circuit, reducing the path deviation and maintaining a safe distance from the tags, but a restriction is imposed on the user of at least always seeing two tags in the camera.

In order to develop the solution in a controlled way and evaluate its behavior in real conditions, a simulation workflow was made for reality. Therefore, it started in the Gazebo Ignition, where the perception and control components have been tested and validated. We then transfer the same workflow to the real LIMO robot, where external noise such as lighting variation, motion blur, and wheel slippage naturally increase the challenge of detection quality for tracking performance. This report therefore documents not only the final system but also the changes that occurred when moving from simulation to a real platform.

The rest of the report presents the requirements of the problem, justifying that it will be useful a tracking system trajectory, and also details the methodology proposed for the method Teach and Replay, both for the part of the validation in simulation and for the implementation in the real case, defines the evaluation metrics and discusses the results obtained with a direct comparison between simulated and real executions.

1.1 Motivation

The motivation to do the work was that it will serve to help students from related areas of programming and robotics at Polytechnique de Clermont-Ferrand to use the system in a practical work, so they can understand how the simple camera perception part, using markers, can be applied directly to mobile robotics, understand the complexity of the transformations of positions used and how, with a simple camera and markers, it is possible to trace a trajectory and, with optimization, arrive at the best possible trajectory.

1.1.1 testesubsub

The motivation to do the work was that it will serve to help students from related areas of programming and robotics at Polytechnique de Clermont-Ferrand to use the system in a practical work, so they can understand how the simple camera perception part, using markers, can be applied directly to mobile robotics, understand the complexity of the transformations of positions used and how, with a simple camera and markers, it is possible to trace a trajectory and, with optimization, arrive at the best possible trajectory.

1.2 Objectives

1.2.1 General objective

1.2.2 Specific objectives

2 Fundamentacao teorica

2.1 Câmera RGB

A câmera RGB embarcada no robô LIMO constitui o principal sensor exteroceptivo utilizado neste projeto, fornecendo uma sequência de imagens 2D a partir das quais são extraídas observações geométricas (cantos de marcadores, contornos e, consequentemente, pose relativa). Do ponto de vista de modelagem, assume-se que o sistema óptico pode ser aproximado por um modelo de câmera pinhole[3] com distorção radial/tangencial, uma hipótese amplamente empregada em visão computacional quando o objetivo é relacionar pontos 3D no espaço a coordenadas 2D na imagem com precisão suficiente para estimação de pose.

Conforme ilustrado na Figura **??**, um ponto 3D no referencial da câmera projeta-se no plano de imagem segundo o modelo pinhole.

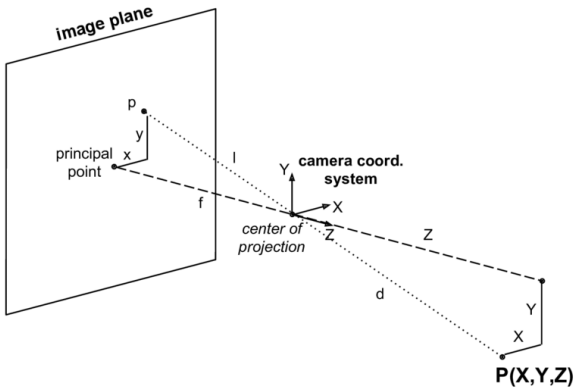

Figure 2.1: Modelo pinhole: projeção de um ponto 3D $P(X, Y, Z)$ no plano de imagem, indicando o centro de projeção e o ponto principal.

Seja um ponto no referencial da câmera $\mathbf{p}_c = [X_c \; Y_c \; Z_c]^\top$, com $Z_c > 0$. Sua projeção ideal[3] (sem distorção) no plano normalizado é dada por

$$x_n = \frac{X_c}{Z_c}, \qquad y_n = \frac{Y_c}{Z_c}. \tag{2.1}$$

A passagem para coordenadas de pixel é determinada pela matriz intrínseca K,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}, \tag{2.2}$$

onde f_x, f_y são as distâncias focais em unidades de pixel e (c_x, c_y) é o centro principal. Na prática, lentes reais introduzem distorção, e a relação entre coordenadas normalizadas ideais (x_n, y_n) e coordenadas distorcidas (x_d, y_d) é frequentemente modelada por termos radiais e tangenciais. Definindo $r^2 = x_n^2 + y_n^2$, um modelo comum é

$$x_d = x_n\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_1 x_n y_n + p_2(r^2 + 2x_n^2), \tag{2.3}$$

$$y_d = y_n\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1(r^2 + 2y_n^2) + 2p_2 x_n y_n, \tag{2.4}$$

e então (u, v) são obtidos substituindo (x_d, y_d) em (**??**). A relevância dessas equações para o presente trabalho decorre do fato de que a estimação de pose baseada em marcadores planares (como AprilTags) depende da precisão na localização de cantos e, portanto, da fidelidade do mapeamento entre o espaço projetivo e o plano de imagem. Mesmo pequenas distorções não modeladas podem introduzir viés sistemático nos cantos detectados, o que se propaga para erros em rotação e translação.

Além da geometria, propriedades físicas do sensor influenciam diretamente a qualidade da percepção. Em particular, ruído de intensidade, compressão e variações de exposição alteram o contraste local utilizado em binarização/adaptação de limiar, enquanto o desfoque de movimento (motion blur) reduz a nitidez de bordas e prejudica a extração de contornos. Um modelo simplificado do motion blur pode ser escrito como uma convolução espacial

$$I_{\text{blur}} = I * h, \tag{2.5}$$

onde I é a imagem ideal e h é um kernel de borramento dependente da velocidade relativa e do tempo de exposição. No contexto de detecção de marcadores, essa degradação afeta principalmente a estimação subpixel dos cantos, reduzindo a repetibilidade das poses obtidas entre frames.

Finalmente, para tarefas de navegação, a câmera deve ser interpretada como um sensor que produz medições com incerteza. Assim, as observações de cantos (e, por consequência, a pose estimada) podem ser vistas como variáveis aleatórias. Um modelo usual assume erro gaussiano aditivo em pixels,

$$\hat{\mathbf{u}}_i = \mathbf{u}_i + \boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \Sigma_u), \tag{2.6}$$

o que justifica, posteriormente, o uso de refinamento por minimização de erro de reprojeção (na estimação de pose) e a adoção de estratégias de controle robustas no seguimento de trajetória.

Em síntese, o modelo pinhole com intrínsecos e distorção ((**??**)–(**??**)), aliado às considerações de ruído e borramento ((**??**)–(**??**)), fornece a base teórica para compreender como a câmera RGB do LIMO influencia a detecção de marcadores e a qualidade das estimativas geométricas utilizadas no restante do sistema.

2.2 AprilTag

AprilTags são marcadores fiduciais planares projetados para fornecer, a partir de uma única imagem, duas informações fundamentais para robótica móvel: (i) uma identificação discreta (ID) e (ii) uma estimativa geométrica de pose 6-DoF do marcador em relação à câmera. A proposta do sistema não é maximizar capacidade de dados, mas sim garantir detecção estável e baixa taxa de falsos positivos sob variações de escala, rotação e iluminação, o que as torna adequadas como *landmarks* artificiais em tarefas de localização e navegação [2]. O detector foi redesenhado no AprilTag 2 com foco explícito em eficiência e robustez computacional [4], e o AprilTag 3 generaliza o uso de layouts e discute critérios de complexidade para preservar confiabilidade mesmo ao ampliar o espaço de padrões possíveis [5].

Para contextualizar a estrutura de uma tag e o papel da borda e das células internas na decodificação, recomenda-se inserir na Figura **??** um exemplo de AprilTag (família e ID) destacando a borda e a região de dados. Em seguida, para conectar a teoria ao pipeline perceptivo, recomenda-se inserir na Figura **??** um fluxograma curto do detector (binarização adaptativa, extração de contornos/quadriláteros, retificação e decodificação), conforme discutido em [4], [5].

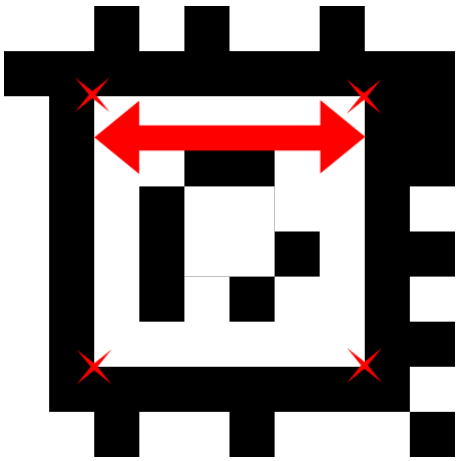

Figure 2.2: Estrutura de uma AprilTag: borda de alto contraste e região de dados (payload) usada na identificação.

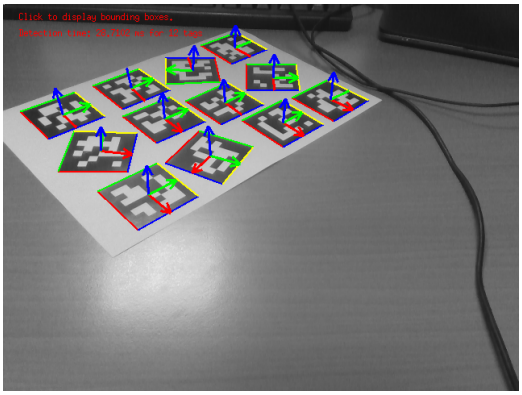

Figure 2.3: Pipeline conceitual de detecção: imagem → limiarização adaptativa → extração de candidatos (quads) → retificação por homografia → leitura de bits e validação do ID.

2.2.1 Decodificação confiável: códigos, rotações e distância de Hamming

A confiabilidade do ID decorre do projeto de famílias de códigos que se mantêm separáveis mesmo quando a tag é observada sob rotações de $0°, 90°, 180°$ e $270°$. Considerando palavras binárias $a, b \in \{0,1\}^n$, a separação entre dois códigos é medida pela distância de Hamming,

$$d_H(a,b) \;=\; \sum_{k=1}^{n} \mathbb{I}\{a_k \neq b_k\}, \tag{2.7}$$

e a geração da família impõe uma distância mínima (incluindo as rotações como equivalências), o que reduz ambiguidades de orientação e diminui a probabilidade de padrões naturais serem aceitos como IDs válidos [2], [4]. Para uma distância mínima d, a capacidade teórica clássica de correção por decisão de vizinho mais próximo é

$$t \;=\; \left\lfloor \frac{d-1}{2} \right\rfloor, \tag{2.8}$$

embora, em aplicações robóticas, uma prática frequente seja rejeitar leituras com erros para diminuir ainda mais falsos positivos [4]. Essa combinação (família bem separada e validação estrita) é a base para tratar AprilTags como marcadores "fiáveis" em ambientes reais.

Figure 2.4: Exemplo de detecção em imagem: quadrilátero estimado, ID decodificado e (opcionalmente) eixos de pose sobrepostos.

2.2.2 Detecção robusta: limiarização adaptativa e extração do quadrilátero

O detector precisa localizar geometricamente a tag antes de decodificar bits. Em cenários reais, um limiar global falha com sombras e iluminação não uniforme; por isso, o AprilTag 2 usa limiarização adaptativa baseada em estatísticas locais (mínimo e máximo) para obter uma binarização que preserve transições preto–branco relevantes [4]. Uma forma canônica de expressar esse princípio é definir, para uma vizinhança $\Omega(x,y)$, um limiar

$$T(x,y) \;=\; \frac{\max\limits_{(i,j)\in\Omega(x,y)} I(i,j) + \min\limits_{(i,j)\in\Omega(x,y)} I(i,j)}{2}, \qquad I_b(x,y) \;=\; \mathbb{I}\{I(x,y) \geq T(x,y)\}, \tag{2.9}$$

onde I é a imagem em tons de cinza e I_b a imagem binária. Em seguida, etapas eficientes de segmentação (por exemplo, por componentes conexas) e agrupamento de bordas permitem construir candidatos a quadriláteros (quads) que satisfaçam restrições geométricas compatíveis com a projeção de um quadrado planar [4]. No AprilTag 3, mantém-se a estrutura do pipeline e discute-se ainda o controle de desempenho por decimação e ajustes voltados a aumentar *recall* em tags pequenas [5].

Para ilustrar a saída perceptiva em nível de imagem (o que o detector realmente "vê"), recomenda-se inserir na Figura **??** um frame real ou do simulador com o contorno do quad e o ID sobreposto; se disponível, incluir também os eixos de pose desenhados sobre a tag (isso ajuda a conectar diretamente com a seção de metodologia/controle).

2.2.3 Por que a pose é estimável: geometria projetiva e homografia

Uma AprilTag é um alvo planar de geometria conhecida; logo, ao observar seus quatro cantos na imagem, a relação entre o plano da tag e o plano de imagem é modelada por uma transformação projetiva (homografia). Assumindo câmera pinhole com matriz intrínseca K, um ponto $\mathbf{X} = [X\ Y\ 0]^\top$ no plano da tag projeta-se como

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} {}^{\text{camera}}R_{\text{tag}} & {}^{\text{camera}}t_{\text{tag}} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{2.10}$$

o que define a homografia

$$H \;=\; K[\, r_1 \;\; r_2 \;\; t\,], \tag{2.11}$$

com r_1, r_2 as duas primeiras colunas de ${}^{\text{camera}}R_{\text{tag}}$ e $t = {}^{\text{camera}}t_{\text{tag}}$ [2]. Assim, a detecção dos cantos fornece correspondências suficientes para estimar H; a partir disso, recupera-se uma estimativa inicial de pose, normalizando escala e impondo ortogonalidade em R.

Em nível prático, uma pose mais precisa é obtida ao refinar (R,t) por minimização do erro de reprojeção dos cantos observados. Denotando por $\mathbf{u}_i$ os cantos medidos na imagem e por $\mathbf{X}_i$ os cantos conhecidos no plano da tag, o refinamento pode ser formulado como

$$\min_{{}^{\text{camera}}R_{\text{tag}} \in SO(3),\, {}^{\text{camera}}t_{\text{tag}}} \sum_{i=1}^{4} \Big\| \mathbf{u}_i - \pi\Big(K\big({}^{\text{camera}}R_{\text{tag}}\mathbf{X}_i + {}^{\text{camera}}t_{\text{tag}} \big) \Big) \Big\|^2, \tag{2.12}$$

Figure 2.5: Geometria da retificação: cantos detectados → estimação de homografia H → patch canônico para leitura do payload.

onde $\pi([x\ y\ z]^\top) = [x/z\ \ y/z]^\top$. Esta equação é a ponte entre percepção e controle: ao reduzir erro de reprojeção, reduz-se a incerteza na pose estimada, o que melhora diretamente a estabilidade do seguimento de trajetória em robótica móvel.

Para reforçar visualmente a ligação entre cantos, retificação e leitura de bits, recomenda-se inserir na Figura **??** um esquema com (a) a tag observada em perspectiva, (b) os quatro cantos detectados, e (c) o patch retificado (vista canônica) usado para amostragem do payload.

2.2.4 AprilTag 3 e confiabilidade sob generalização de layouts

O AprilTag 3 estende o sistema ao permitir layouts mais flexíveis, o que amplia possibilidades de desenho (incluindo regiões ignoradas e diferentes distribuições de bits), mas torna ainda mais importante controlar falsos positivos. O trabalho discute critérios de complexidade para garantir que os padrões gerados sejam improváveis em imagens naturais; entre eles, avalia-se a energia do modelo de Ising aplicada à imagem binária do layout [5]. Em termos discretos, uma escrita típica dessa métrica é

$$E \;=\; \sum_{i,j} \Big(\mathbb{I}\{I(i,j) \neq I(i,j+1)\} + \mathbb{I}\{I(i,j) \neq I(i+1,j)\} \Big), \tag{2.13}$$

que corresponde ao comprimento total de fronteiras preto–branco. A intuição é que padrões com estrutura local "muito regular" tendem a ocorrer com maior probabilidade em cenas artificiais/-naturais; logo, impor complexidade adequada reduz a chance de um padrão aleatório passar pelo processo de validação e ser aceito como tag [5].

3 Otmizacao e trajetoria

Apos um visualizaçao em que um camera se move em uma trajetoria que se paraece com incosaedro, durante a visualizaçao foi garantido que a todo momento a camera ve pelo menos duas tags;

A primeira etapa a realizar é fazer mapaemento do nosso ambiente. Sendo assim, definir a primeira tag referencia o calculadas tranformaçoes relativas para as tags subsequentes. dado que eu sei onde todas as tags estao localizada posiçao e orientaçao, eu posso localizar o meu robo no ambiente.

A problemantica é que o robo nao consegue ver todas as tags ao mesmo tempo e também, dado que o robo ve as tags em sequencia pequenos erros de detecçao da posicao do robo sao acumulados a medida em que o robo se move

Logo o mapaemento deve ser em caideia fazendo as transformada de referencial a partir de uma tag referencia, ou a tag mundo.

A camera é colocada como pose de referencia e recebe a pose relativa da tagA, primeira tag vista, ao mesmo tempo, é possivel ver a tagB, Assim atraves destas poses é possivel cacula a pose da tag B em relaçao a tagA:

$$T_{A\to B} = T_{Cam\to A}^{-1} * T_{Cam\to B} \tag{3.1}$$

- $T_{A\to B} = (R_A, \boldsymbol{t}_A)$: Pose da Tag B em relaçao a A.
- $T_{Cam\to A} = (R_A, \boldsymbol{t}_A)$: Pose da Tag A vista pela Câmera.
- $T_{Cam\to B} = (R_B, \boldsymbol{t}_B)$: Pose da Tag B vista pela Câmera.
- R: Matriz de rotaçao
- t: Vetor de posiçao

A tranformaçao de inveresa é caculada da seguinte forma:

$$T^{-1} = (R^T, -R^T\boldsymbol{t}) \tag{3.2}$$

dada que o as orientaçoes das tags sao dadas em quartenion, é feita a conversao para matrizes de rotacao:

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.3}$$

Assim é possivel calcular a rotaçao relativa(R_{rel}):

$$R_{rel} = R_A^T R_B \tag{3.4}$$

Dado que a detecao da tag também nos fornece a posiçao, assim também é possivel calcular: a translaçao relativa ($\boldsymbol{t}_{rel}$):

$$\boldsymbol{t}_{rel} = R_A^T(\boldsymbol{t}_B - \boldsymbol{t}_A) \tag{3.5}$$

Assim

$$T_{A\to B} = (R_{rel}, \boldsymbol{t}_{rel}) \tag{3.6}$$

também é possivel realisar o calculo atravez de matrizes de transformação homogenea:

$$\begin{bmatrix} R_{rel} & \boldsymbol{t}_{rel} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} R_A & \boldsymbol{t}_A \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}^{-1} \begin{bmatrix} R_B & \boldsymbol{t}_B \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{3.7}$$

Assim, dado que o nosso ambiente é conhecido, ou seja a pose das todas as trafomaçoes das tags sao conhecidas, de igual modo é possivel localizar a pose do robo.

A partir da detecçao da pose da tag em realçao a camera é feito uma tranformaçao inversa da pose da tag até a camera para encontrar qual a pose da comaera em ralaçao a tag

$$T_{W,C} = T_{W,T} * (T_{C,T})^{-1} \tag{3.8}$$

Em que:

$T_{W,T}$ é a posição da Tag no seu mapa (onde a Tag 0 é a origem). $T_{C,T}$ é a detecção (a Tag vista pela Câmera). $(T_{C,T})^{-1}$ é a pose da Câmera em relação à Tag. (3.9)

Se a tag que a camera vé nao é a tag zero, mas dado que o ambiente foi mapeado, sabemos a pose desta tag em relaçao a tag0 e asse por meio de tranformaçao inversa é possivel calcular a pose da camera.

Para pose da camera quanto mais tags forem vista mais precisao temos para a pose da camera, visto que podemos calcular a media da pose da camera no mundo em ralaçao a cada tag. Por um metodo chamado SLERP, que envolve a converçao da rotaçao para quartenion.

As matrizes de rotação $\mathbf{R} \in SO(3)$ obtidas do sistema de visão são convertidas para quaternions unitários $\mathbf{q} = [w, x, y, z]^T$. Dada uma matriz $\mathbf{R}$ com elementos r_{ij}, o componente escalar w é calculado a partir do traço da matriz ($Tr(\mathbf{R})$):

$$w = \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}} \tag{3.10}$$

Os demais componentes (x, y, z) são derivados das diferenças entre os elementos simétricos da matriz:

$$x = \frac{r_{32} - r_{23}}{4w}, \quad y = \frac{r_{13} - r_{31}}{4w}, \quad z = \frac{r_{21} - r_{12}}{4w} \tag{3.11}$$

3.0.1 Média da Translação (Posição)

A posição final da câmera é calculada através da média aritmética simples das estimativas de posição $\mathbf{p}_1$ e $\mathbf{p}_2$:

$$\mathbf{p}_{avg} = \frac{\mathbf{p}_1 + \mathbf{p}_2}{2} \tag{3.12}$$

3.0.2 Interpolação Esférica Linear (SLERP)

Para a rotação, utiliza-se o *Spherical Linear Interpolation* (SLERP), que realiza a interpolação ao longo do arco da hiperesfera unitária. Para duas rotações $\mathbf{q}_1$ e $\mathbf{q}_2$ com fator de peso $t = 0.5$:

A. **Caminho Curto:** Calcula-se o produto escalar $\cos(\theta) = \mathbf{q}_1 \cdot \mathbf{q}_2$. Se $\cos(\theta) < 0$, inverte-se o sinal de um dos quaternions ($\mathbf{q}_2 = -\mathbf{q}_2$) para garantir o caminho mais curto.

B. **Cálculo do SLERP:**

$$\mathbf{q}_{avg} = \frac{\sin((1-t)\theta)}{\sin(\theta)}\mathbf{q}_1 + \frac{\sin(t\theta)}{\sin(\theta)}\mathbf{q}_2 \tag{3.13}$$

3.1 1. Otimização do Grafo de Poses

O objetivo é encontrar as poses globais das tags T_w^i que minimizam o erro de consistência com as medidas relativas Z_{ij} obtidas pela mediana das observações.

3.1.1 Vetor de Estado (Por que 6 parâmetros e não 7?)

Embora uma pose rígida seja composta por translação (3) e quaternion (4), o vetor de otimização $\mathbf{x}$ utiliza 6 parâmetros por tag para evitar superparametrização e a restrição de norma unitária do quaternion ($\|q\| = 1$), que tornaria o Jacobiano singular.

A rotação é parametrizada no espaço tangente $\mathfrak{so}(3)$ (Vetor de Rotação ou *Axis-Angle*):

$$\mathbf{x} = [\mathbf{v}_1^T, \mathbf{t}_1^T, \ldots, \mathbf{v}_N^T, \mathbf{t}_N^T]^T \tag{3.14}$$

Onde $\mathbf{v}_i \in \mathbb{R}^3$. Durante a otimização, $\mathbf{v}_i$ é convertido para matriz de rotação via exponencial map $R_i = \exp([\mathbf{v}_i]_\times)$ para cálculo do erro.

3.1.2 Conversão: Rotação ↔ Vetor (Exponencial e Logarítmico)

Para transitar entre o vetor de estado $\mathbf{v}$ e as matrizes de rotação R, utiliza-se a Fórmula de Rodrigues e o logaritmo de matrizes.

3.1.3 2. Matriz → Vetor de Rotação (Log Map)

Usada para converter a matriz de erro em um vetor residual minimizável.

$$\theta = \arccos\left(\frac{\mathrm{Tr}(R)-1}{2}\right), \quad \mathbf{u} = \frac{1}{2\sin(\theta)}\left[r_{32}-r_{23}\; r_{13}-r_{31}\; r_{21}-r_{12}\right] \tag{3.15}$$

3.1.4 3. Quaternion → Vetor de Rotação

Usada na inicialização para converter o chute inicial (quaternion) em estado do solver.

$$\theta = 2\arccos(w), \quad \mathbf{v} = \theta \cdot \frac{[x,y,z]^T}{\sin(\theta/2)} \tag{3.16}$$

3.1.5 4. Vetor de Rotação → Quaternion

Usada na saída para salvar o resultado final.

$$w = \cos(\theta/2), \quad [x,y,z]^T = \sin(\theta/2)\cdot\mathbf{u} \tag{3.17}$$

3.1.6 Função de Resíduo e Seus Termos

Para uma aresta conectando a Tag i à Tag j com medição $Z_{ij} = (\hat{R}_{ij}, \hat{\mathbf{t}}_{ij})$, o resíduo $\mathbf{r}_{ij}$ é composto por:

→ **Erro de Posição:** Diferença vetorial entre a posição da tag j projetada no referencial de i e a medição.

$$\mathbf{r}_{pos} = \underbrace{R_i^T(\mathbf{t}_j - \mathbf{t}_i)}_{\text{Predição Local}} - \underbrace{\hat{\mathbf{t}}_{ij}}_{\text{Medição}} \tag{3.18}$$

- R_i^T: Rotaciona o vetor de diferença global $(\mathbf{t}_j - \mathbf{t}_i)$ para o referencial local da Tag i, permitindo comparação direta com a medição do sensor $\hat{\mathbf{t}}_{ij}$.

→ **Erro de Rotação:** Desvio angular necessário para alinhar a rotação estimada com a medida.

$$\mathbf{r}_{rot} = \lambda \cdot \mathrm{rotvec}(\underbrace{(\hat{R}_{ij})^T R_i^T R_j}_{\text{Matriz de Erro}}) \tag{3.19}$$

- $(\hat{R}_{ij})^T R_i^T R_j$: Produto que resulta na matriz identidade se o alinhamento for perfeito.

- λ: Fator de ponderação ($\texttt{rotation}_{w}eight)queequilibraamagnitudedoerroangular(radianos)comoerrolinear(m$

3.1.7 Função de Custo Robusta (Fórmula de Huber)

Minimiza-se a função de custo global utilizando a Perda de Huber ρ_δ para rejeição de *outliers* (erros grosseiros):

$$F(\mathbf{x}) = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} \rho_\delta(w_{ij}(\|\mathbf{r}_{pos}\|^2 + \|\mathbf{r}_{rot}\|^2)) \tag{3.20}$$

A função ρ_δ evita que erros grandes "puxem" a solução, comportando-se linearmente para erros acima do limiar δ:

$$\rho_\delta(s) = \begin{cases} s & \text{se } s \leq \delta^2 \text{ (Quadrático/Gaussiano) } 2\delta(\sqrt{s} - \frac{\delta}{2}) \\ \text{se } s > \delta^2 \text{ (Linear/Robust)} \end{cases} \tag{3.21}$$

O termo δ (`huber`$_s cale) de fine o limite entre um erro aceitvel (rudo gaussiano) e um$ `outlier`.

3.2 2. Processo de Atualização (Solução Numérica)

A minimização é realizada iterativamente pelo algoritmo *Trust Region Reflective*. Em cada iteração k:

A. Jacobiano (J): Calcula-se a matriz de derivadas parciais $J = \frac{\partial \mathbf{r}}{\partial \mathbf{x}}$. Esta matriz indica a sensibilidade do erro em relação a cada variável de estado.

B. Passo ($\Delta\mathbf{x}$): Resolve-se o sistema linear aproximado para encontrar o passo de ajuste, onde μ é o fator de amortecimento (*damping*):

$$(J^T J + \mu I)\Delta\mathbf{x} = -J^T \mathbf{r} \tag{3.22}$$

C. Atualização: As variáveis são atualizadas somando o passo ao estado atual:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x} \tag{3.23}$$

Como usamos o vetor de rotação $\mathbf{v}$, a atualização da rotação é uma simples soma vetorial no espaço $\mathbb{R}^3$, livre de restrições de norma.

4 Metodologia

5 Results

6 Discussion

7 Conclusion

References

[1] P. Nourizadeh, M. Milford, and T. Fischer, *Teach and repeat navigation: A robust control approach*, 2024. arXiv: 2309.15405 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2309.15405.

[2] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2011, pp. 3400–3407.

[3] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second. Cambridge University Press, ISBN: 0521540518, 2004.

[4] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016.

[5] M. Krogius, A. Haggenmiller, and E. Olson, "Flexible layouts for fiducial tags," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2019.