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 . . . **3**

1.1 Motivation . . . 3

1.1.1 testesubsub . . . 4

1.2 Objectives . . . 4

1.2.1 General objective . . . 4

1.2.2 Specific objectives . . . 4

2 Fundamentacao teorica . . . **5**

2.1 Câmera RGB . . . 5

2.2 AprilTag . . . 6

2.2.1 Decodificação confiável: códigos, rotações e distância de Hamming . . . 7

2.2.2 Detecção robusta: limiarização adaptativa e extração do quadrilátero . . . 8

2.2.3 Por que a pose é estimável: geometria projetiva e homografia . . . 8

2.2.4 AprilTag 3 e confiabilidade sob generalização de layouts . . . 9

3 Theoretical Background . . . **10**

3.1 Conversions and Rotation Calculation . . . 10

3.1.1 Conversion: Rotation $\leftrightarrow$ Vector (Exponential and Logarithmic) . . . 10

3.1.2 2. Matrix $\rightarrow$ Rotation Vector (Log Map) . . . 10

3.1.3 3. Quaternion $\rightarrow$ Rotation Vector . . . 10

3.1.4 4. Rotation Vector $\rightarrow$ Quaternion . . . 10

4 Methodology . . . **11**

4.1 Trajectory Optimization and Mapping . . . 11

4.2 Relative Transformations . . . 11

4.3 Robot Localization and Pose Averaging . . . 11

4.4 1. Pose Graph Optimization . . . 12

4.4.1 State Vector (Why 6 parameters and not 7?) . . . 12

4.5 Tag Optimization with Levenberg-Marquardt . . . 12

4.6 Residual per Edge . . . 12

4.7 Global Cost Function . . . 13

4.8 Numerical Update . . . 13

4.9 Camera Trajectory Optimization . . . 13

4.10 LM Update per Frame . . . 14

4.11 Numerical Flow Summary . . . 14

5 Results **15**
5.1 Ideal Scenario 15
5.2 Robustness Analysis 16

6 Metodologia **17**

7 Results **18**

8 Discussion **19**

9 Conclusion **20**

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**nourizadeh2024teachrepeatnavigationrobust**, 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 **olson2011tags** 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**Hartley2004** 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 2.1, um ponto 3D no referencial da câmera projeta-se no plano de imagem segundo o modelo pinhole.

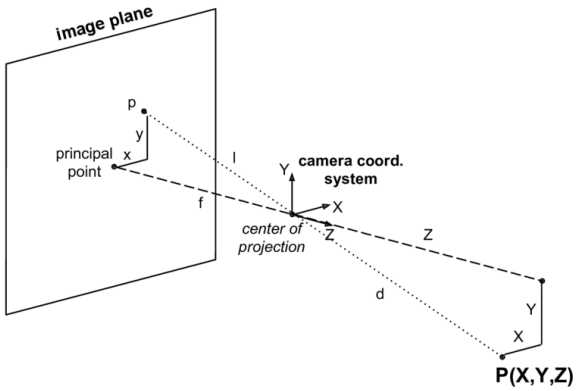

Figure 2.1: Modelo pinhole: projeção de um ponto 3D $P(X, Y, Z)$ no plano de imagem, indicando o centro de projeção e o ponto principal.

Seja um ponto no referencial da câmera $\mathbf{p}_c = [X_c \;\; Y_c \;\; Z_c]^\top$, com $Z_c > 0$. Sua projeção ideal**Hartley2004** (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 (2.2). 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 ((2.1)–(2.4)), aliado às considerações de ruído e borramento ((2.5)–(2.6)), 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 **olson2011tags**. O detector foi redesenhado no AprilTag 2 com foco explícito em eficiência e robustez computacional **wang2016iros**, 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 **krogius2019iros**.

Para contextualizar a estrutura de uma tag e o papel da borda e das células internas na decodificação, recomenda-se inserir na Figura 2.2 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 2.3 um fluxograma curto do detector (binarização adaptativa, extração de contornos/quadriláteros, retificação e decodificação), conforme discutido em **wang2016iros**, **krogius2019iros**.

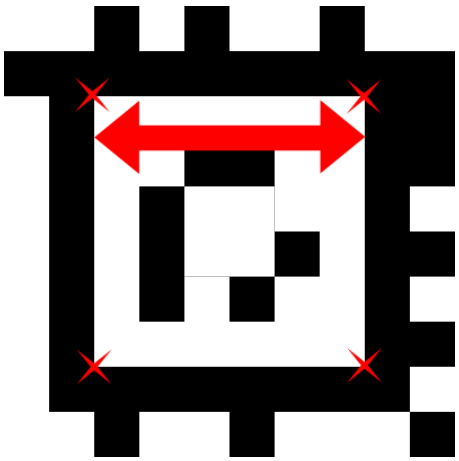

Figure 2.2: Estrutura de uma AprilTag: borda de alto contraste e região de dados (payload) usada na identificação.

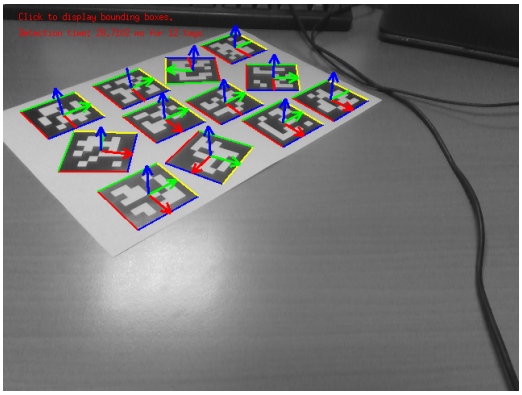

Figure 2.3: Pipeline conceitual de detecção: imagem → limiarização adaptativa → extração de candidatos (quads) → retificação por homografia → leitura de bits e validação do ID.

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 **olson2011tags**, **wang2016iros**. 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 **wang2016iros**. 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 **wang2016iros**. 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 **wang2016iros**. 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 **krogius2019iros**.

Para ilustrar a saída perceptiva em nível de imagem (o que o detector realmente "vê"), recomenda-se inserir na Figura 2.4 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 $t = {}^{\text{camera}}t_{\text{tag}}$ **olson2011tags**. 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 2.5 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 **krogius2019iros**. 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 **krogius2019iros**.

3 Theoretical Background

3.1 Conversions and Rotation Calculation

Since tag orientations are provided as unit quaternions $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$, they are converted in rotation matrices $R \in SO(3)$ for homogeneous matrix composition, **diebel2006representing**.

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{3.1}$$

The relative rotation (R_{rel}) and relative translation ($\boldsymbol{t}_{rel}$) are computed how it follows, **lynch2017modern**:

$$R_{rel} = R_A^T R_B, \quad \boldsymbol{t}_{rel} = R_A^T(\boldsymbol{t}_B - \boldsymbol{t}_A) \tag{3.2}$$

3.1.1 Conversion: Rotation $\leftrightarrow$ Vector (Exponential and Logarithmic)

To transition between the state vector $\mathbf{v}$ and rotation matrices R, Rodrigues' Formula and the matrix logarithm are used, **barfoot2024state**.

3.1.2 2. Matrix $\rightarrow$ Rotation Vector (Log Map)

Used to convert the error matrix into a minimizable residual vector, **barfoot2024state**.

$$\theta = \arccos\left(\frac{\mathrm{Tr}(R) - 1}{2}\right), \quad \mathbf{u} = \frac{1}{2\sin(\theta)}\begin{bmatrix} r_{32} - r_{23} & r_{13} - r_{31} & r_{21} - r_{12} \end{bmatrix} \tag{3.3}$$

3.1.3 3. Quaternion $\rightarrow$ Rotation Vector

Used during initialization to convert the initial guess (quaternion) into the solver state, **diebel2006representing**.

$$\theta = 2\arccos(w), \quad \mathbf{v} = \theta \cdot \frac{[x, y, z]^T}{\sin(\theta/2)} \tag{3.4}$$

3.1.4 4. Rotation Vector $\rightarrow$ Quaternion

Used at the output to save the final result, **diebel2006representing**.

$$w = \cos(\theta/2), \quad [x, y, z]^T = \sin(\theta/2) \cdot \mathbf{u} \tag{3.5}$$

4 Methodology

4.1 Trajectory Optimization and Mapping

After a visualization session where the camera moves in a trajectory resembling an icosahedron, it was ensured that the camera observes at least two tags at any given time.

The first stage is environment mapping. By defining the first tag as the world reference, relative transformations are calculated for all subsequent tags. Once the position and orientation of all tags are established, the robot can be localized within the environment, **olson2011apriltag**.

The primary challenge is that the robot cannot see all tags simultaneously. Furthermore, since tags are detected sequentially, small estimation errors in the robot's pose accumulate as it moves, leading to *drift*, **barfoot2024state**. Therefore, mapping must be performed as a chain of transformations relatives to a reference tag (World Tag).

4.2 Relative Transformations

The camera serves as a temporary reference frame. When Tag_A and Tag_B are visible simultaneously, the pose of Tag_B relative to Tag_A is calculated through the following composition, **lynch2017modern**:

$$T_{A\to B} = T_{Cam\to A}^{-1} \cdot T_{Cam\to B} \tag{4.1}$$

Where:

- $T_{A\to B} = (R_{rel}, \boldsymbol{t}_{rel})$: Pose of Tag_B relative to Tag_A.
- $T_{Cam\to A} = (R_A, \boldsymbol{t}_A)$: Pose of Tag_A in the camera frame.
- $T_{Cam\to B} = (R_B, \boldsymbol{t}_B)$: Pose of Tag_B in the camera frame.

The inverse transformation is defined as, **lynch2017modern**:

$$T^{-1} = (R^T, -R^T\boldsymbol{t}) \tag{4.2}$$

4.3 Robot Localization and Pose Averaging

Given a known map, the camera's pose in the world frame ($T_{W,C}$) is estimated from a detected tag, typically via a PnP formulation solved efficiently by EPnP, **lepetit2009epnp**:

$$T_{W,C} = T_{W,T} \cdot (T_{C,T})^{-1} \tag{4.3}$$

To improve precision, when multiple tags are visible, the estimates are fused. The translation is averaged arithmetically, while the rotation is fused using *Spherical Linear Interpolation* (SLERP), **shoemake1985slerp**.

4.4 1. Pose Graph Optimization

The optimization for the tags aims to find the global poses T_w^i that minimize the error between the calculated edges between tags, using tag0 as the reference, **barfoot2024state**.

Given that when the robot moves, measurements of relative tag poses with respect to the camera pose are made at each frame. Taking into account frames where the robot sees more than one tag, it is possible to calculate the edges at each pose. Since we have multiple measurements, we take an initial value before optimization.

4.4.1 State Vector (Why 6 parameters and not 7?)

In the optimization calculation, although the position plus the quaternion has 7 parameters, we use only 6 because we convert the Quaternion into a rotation vector to avoid estimating values during optimization that do not satisfy the quaternion norm ($\|q\| = 1$), **barfoot2024state**.

Rotation is parameterized in the tangent space $\mathfrak{so}(3)$ (Rotation Vector or *Axis-Angle*):

$$\mathbf{x} = [\mathbf{v}_1^T, \mathbf{t}_1^T, \ldots, \mathbf{v}_N^T, \mathbf{t}_N^T]^T \tag{4.4}$$

After prediction, the rotation vector is converted into a rotation matrix for error calculation.

4.5 Tag Optimization with Levenberg-Marquardt

Optimization is based on a pose graph that connects tag poses pairwise, $\mathbf{T}_i$ and $\mathbf{T}_j$. For each tag pair, we have a relative measurement $\hat{\mathbf{T}}_{ij} = (\hat{\mathbf{R}}_{ij}, \hat{\mathbf{t}}_{ij})$, **barfoot2024state**.

4.6 Residual per Edge

Thus, it is possible to calculate the position error by subtracting the predicted value (the difference between tag poses) and the measured value, **barfoot2024state**. Given global poses $\mathbf{T}_i = (\mathbf{R}_i, \mathbf{t}_i)$ and $\mathbf{T}_j = (\mathbf{R}_j, \mathbf{t}_j)$, is calculated the relative prediction.

To calculate the error between one tag and another, we have two errors: the translation error and the rotation error. The position error is calculated by subtracting the predicted difference between tag translations. For the rotation error, the relative orientation between two tags is calculated; that is, by multiplying the conjugate of rotation RtagA by RtagB and the measurement. If the orientations overlap, the quaternion will have element w = 1 and all others = 0, **barfoot2024state**.

$$\mathbf{R}_{ij} = \mathbf{R}_i^\top \mathbf{R}_j, \qquad \mathbf{t}_{ij} = \mathbf{R}_i^\top (\mathbf{t}_j - \mathbf{t}_i). \tag{4.5}$$

The stacked position+rotation residual is

$$\mathbf{r}_{ij} = \begin{bmatrix} \mathbf{t}_{ij} - \hat{\mathbf{t}}_{ij} \\ \mathrm{Log}\left(\hat{\mathbf{R}}_{ij}^\top \mathbf{R}_{ij}\right) \cdot \alpha \end{bmatrix} \in \mathbb{R}^6, \tag{4.6}$$

Where α is the relative weight of the rotation (e.g., $\alpha =$ `rotation_weight`), **barfoot2024state**.

4.7 Global Cost Function

With the residuals calculated for the set of edges $\mathcal{E}$ and statistical weights w_{ij}, we use the linear loss cost function that accumulates the error values, **barfoot2024state**.

$$J(\mathbf{x}) = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} w_{ij}^2 \, \rho\left(\left\|\mathbf{r}_{ij}(\mathbf{x})\right\|_2\right), \tag{4.7}$$

Where $\mathbf{x}$ are the accumulated values for the tag variables and ρ is **linear**. Robustness depends of explicit pruning of high-residual edges.

4.8 Numerical Update

Minimization is performed using the Levenberg-Marquardt algorithm in its damped least-squares form (trust-region perspective), matching the update in **turkulainen20243dgs**.

This method calculates the Jacobian of the cost function. The value of λ defines whether the error response behaves like the Gauss-Newton method (fast convergence) or gradient descent (slow convergence). The value of λ is calculated internally by the library according to the Jacobian and the initial parameter update. As in **turkulainen20243dgs**, the update function has the form:

$$\left(\mathbf{J}^\top \mathbf{J} + \lambda \mathbf{I}\right) \Delta\mathbf{x} = -\mathbf{J}^\top \mathbf{r}, \tag{4.8}$$

where $\mathbf{J}$ is the Jacobian of constraints, and $\lambda \geq 0$ is the damping factor.

4.9 Camera Trajectory Optimization

Tag optimization works similarly, but here we already know the tag map, so we only need to calculate the camera pose in each frame based on this map,

$\{\mathbf{p}_{k\ell}^{(c)}\}$ (positions of tag ℓ in the camera frame). The camera pose $\mathbf{T}_{wk} = (\mathbf{R}_{wk}, \mathbf{t}_{wk})$ maps, **Hartley2004**

$$\mathbf{p}_{k\ell}^{(w)} = \mathbf{R}_{wk} \, \mathbf{p}_{k\ell}^{(c)} + \mathbf{t}_{wk}. \tag{4.9}$$

We only have a change in the cost function, which is calculated by the difference between poses. However, here we also have 6 calculated parameters, so the position residual is, **barfoot2024state**

$$\mathbf{r}_{k\ell} = \mathbf{p}_{k\ell}^{(w)} - \mathbf{p}_{\ell}^{(w)}. \tag{4.10}$$

The frame cost function is, **barfoot2024state**

$$J_k(\boldsymbol{\theta}_{wk}, \mathbf{t}_{wk}) = \frac{1}{2} \sum_{\ell \in \mathcal{O}_k} \|\mathbf{r}_{k\ell}\|_2^2, \tag{4.11}$$

Where $\mathcal{O}_k$ is defined according to the number of visible tags in frame k.

4.10 LM Update per Frame

For each frame, LM solves the same equation used previously to estimate the graphs, following **turkulainen20243dgs**:

$$\left(\mathbf{J}_k^\top \mathbf{J}_k + \lambda \mathbf{I}\right) \Delta \mathbf{x}_k = -\mathbf{J}_k^\top \mathbf{r}_k, \tag{4.12}$$

with $\mathbf{x}_k = [\boldsymbol{\theta}_{wk}, \mathbf{t}_{wk}]$ and exponential update for rotation.

4.11 Numerical Flow Summary

The iterative process and processing steps of the optimization algorithm are illustrated in Figure 4.1. This flowchart details everything from edge construction based on relative observations to solving the PnP for the camera trajectory.

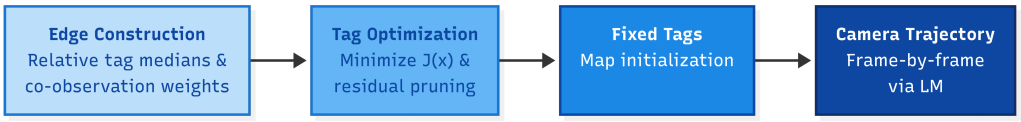

Figure 4.1: Visual representation of the numerical flow: edge construction, initialization, tag pose optimization, and camera trajectory calculation.

5 Results

The algorithm validation was performed using synthetic datas, which allows for direct comparison with Ground Truth to quantify optimization precision and robustness.

5.1 Ideal Scenario

Initially, the system was tested with perfect detections, free of drift. For this data, the residual error was null, confirming the geometric consistency of the algorithm. The metrics are presented in Table 5.1.

Metric	Mean Error	Minimum	Maximum
Camera Trajectory	0.0000 m/0.00°	0.0000 m/0.00°	0.0000 m/0.00°
Tag Poses	0.0000 m/0.00°	0.0000 m/0.00°	0.0000 m/0.00°

Table 5.1: Residual errors in the ideal scenario (no noise).

Figure 5.1 presents the visual results for this scenario. On the left, the alignment of estimated tags with the real scenario is observed, showing that the tags were perfectly aligned as suggested by the table, without errors. On the right, the optimized trajectory is superimposed on the reference circular trajectory, verifying that the trajectory was perfectly recovered without errors.

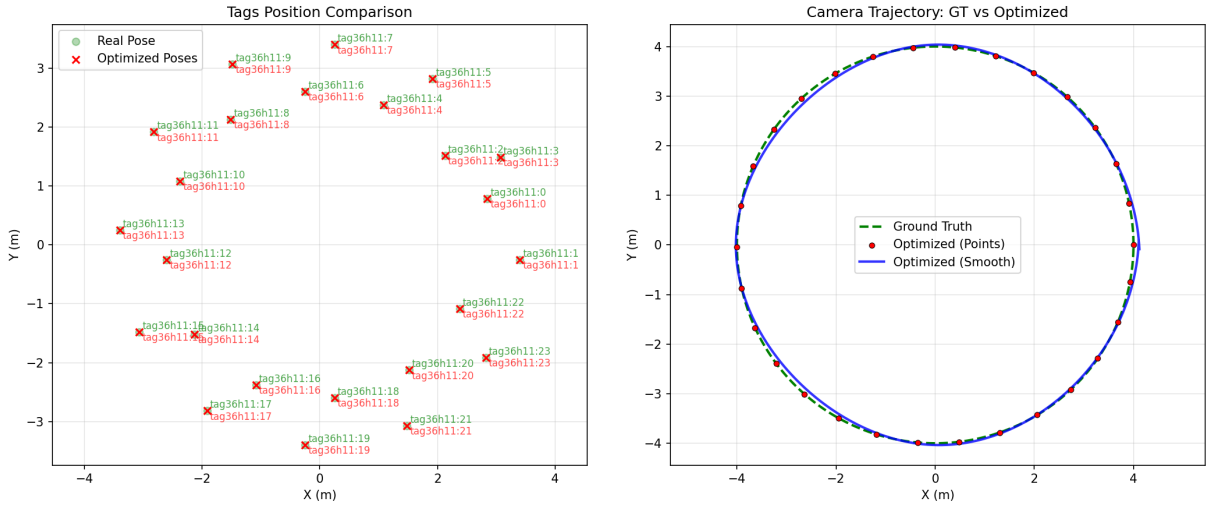

Figure 5.1: Results for the ideal scenario: on the left, the tag map (Estimated vs. Real) and on the right, the camera trajectory.

5.2 Robustness Analysis

To evaluate robustness, Gaussian noise of 10 cm was injected into the position measurements. The optimization algorithm demonstrated high efficacy in filtering this noise, as shown in the data presented in Table 5.2.

Metric	Mean Error	Minimum	Maximum
Camera Trajectory	0.0584 m/0.97°	0.0126 m/0.11°	0.1485 m/2.67°
Tag Poses	0.0364 m/0.73°	0.0156 m/0.73°	0.0583 m/0.73°

Table 5.2: Error statistics under 10 cm Gaussian noise.

Figure 5.2 illustrates the visual comparison for this noisy scenario, demonstrating that the system can recover the map structure and trajectory with centimeter-level precision even under significant uncertainty. Even with 10 cm Gaussian noise, the algorithm proved robust: it reduced tag pose error from $\approx$ 10 cm to about 3 cm and camera pose error to roughly 0.6 cm. The visualization confirms that tags remain well localized and the trajectory is effectively recovered despite the small residual error.

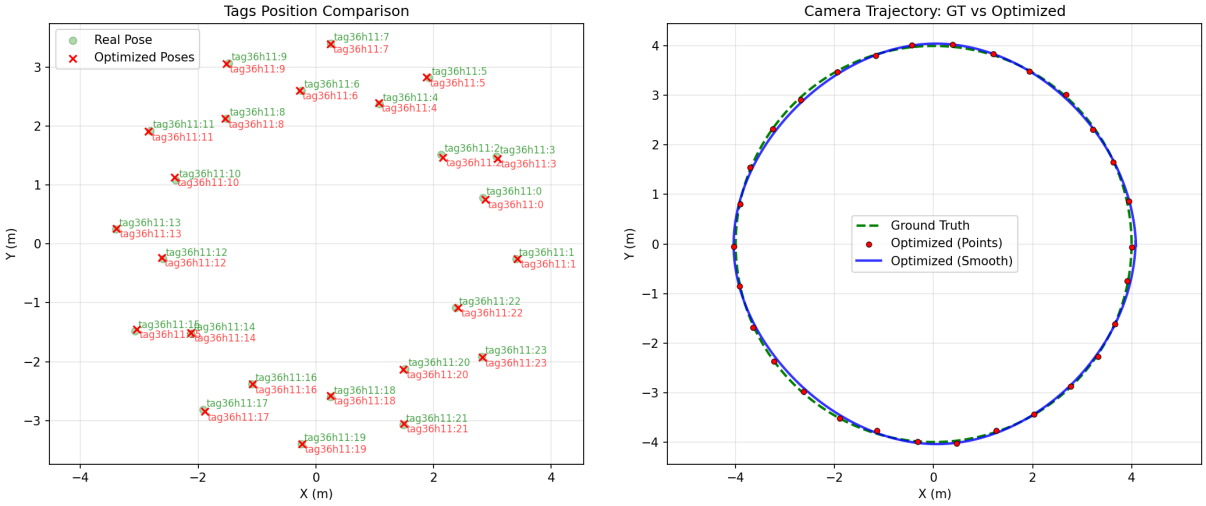

Figure 5.2: Results under 10 cm Gaussian noise: on the left, map recovery and on the right, the trajectory.

6 Metodologia

7 Results

8 Discussion

9 Conclusion