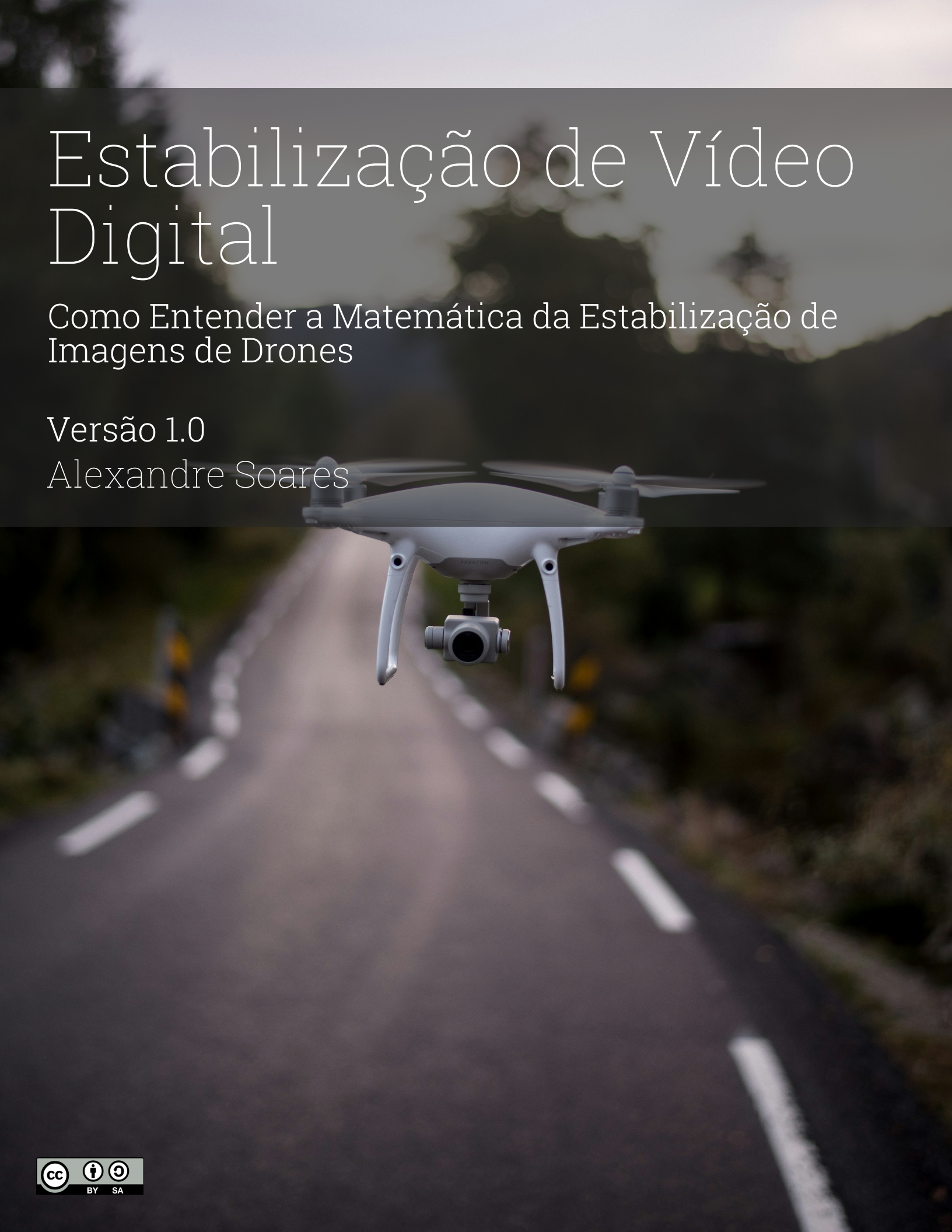
Estabilização de Vídeo Digital

Como Entender a Matemática da Estabilização de Imagens de Drones

Versão 1.0

Alexandre Soares



Estabilização de Vídeo Digital

Como Entender a Matemática da Estabilização de Imagens de Drones

por

Alexandre Soares

Notas Sobre o Estudo

Todos os programas fonte e dados utilizados para a elaboração deste estudo, bem como o PDF deste material, estão disponíveis para download no repositório GitHub, no endereço: https://github.com/Alxsoa/Artigos.

Conteúdo

1

Motivação do Trabalho

1.1. Objetivo do Estudo

Movimentos indesejados da câmera de um drone, causados por condições climáticas, vibrações de veículos ou instabilidades mecânicas, introduzem artefatos visuais que degradam significativamente a qualidade perceptual do vídeo. A estabilização digital oferece uma solução computacional para este problema, evitando a necessidade de estabilizadores mecânicos custosos.

Este estudo apresenta uma análise matemática completa e rigorosa de um algoritmo de estabilização de vídeo digital implementado em Python utilizando OpenCV.

O sistema emprega técnicas avançadas de visão computacional, processamento de sinais digitais e álgebra linear para corrigir movimentos indesejados em sequências de vídeo. A abordagem baseia-se principalmente em optical flow Lucas-Kanade [1], transformações geométricas afins e filtragem temporal por média móvel.

Este documento fornece fundamentação teórica completa, derivações matemáticas detalhadas, análise de estabilidade e robustez, além de discussões sobre complexidade computacional e otimização de parâmetros. O estudo serve como referência tanto para aplicações práticas quanto para desenvolvimentos acadêmicos na área de estabilização de vídeo.

2

Fundamentação Teórica

2.1. Formulação Matemática do Problema

Seja $\{I_t\}_{t=1}^{N}$ uma sequência de N frames de vídeo, onde cada imagem $I_t : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$ representa uma imagem colorida definida sobre um domínio espacial Ω. A estabilização de vídeo tem como objetivo compensar movimentos indesejados, como vibrações ou deslocamentos bruscos, presentes nos frames capturados, especialmente em aplicações como vídeos embarcados em drones.

O problema pode ser formalizado como a busca por um conjunto de transformações $\{T_t\}_{t=1}^{N-1}$ aplicadas aos frames originais, de modo que a nova sequência corrigida $\{I'_t\}_{t=1}^{N}$ apresente menor instabilidade visual. Cada imagem estabilizada é obtida por meio da aplicação inversa da transformação estimada, ou seja:

$$I'_t = T_t^{-1}(I_t) \tag{2.1}$$

Esse processo visa minimizar uma medida global de oscilação, como a variação da posição de pontos de interesse ao longo do tempo.

O movimento entre frames consecutivos pode ser modelado da seguinte forma:

$$I_{t+1}(\mathbf{x}) = I_t(T_t(\mathbf{x})) + \eta_t(\mathbf{x}) \tag{2.2}$$

onde $\mathbf{x} = (x, y)^T$ representa as coordenadas espaciais, T_t é a transformação geométrica aplicada ao frame t, e η_t é um termo que representa ruído e componentes da cena não modelados diretamente pela transformação.

Com base nesse modelo, a estabilização de vídeo pode ser entendida como um problema de estimativa de movimento seguido de correção geométrica. A eficácia da estabilização depende da escolha do modelo de transformação T_t e da robustez frente ao ruído η_t.

Métodos tradicionais baseiam-se em fluxo óptico ou transformações afins, enquanto abordagens mais recentes utilizam redes neurais profundas para inferir a estabilização de maneira aprendida e adaptativa, especialmente útil em cenários desafiadores como filmagens aéreas.

3

Algoritmo Lucas-Kanade

3.1. Explicando o Conceito de Fluxo Óptico

O conceito de *optical flow* (fluxo óptico) baseia-se no princípio da **constância de brilho** (*brightness constancy constraint*), que assume que a intensidade observada de um ponto da cena permanece aproximadamente constante enquanto ele se desloca entre frames consecutivos. Essa suposição pode ser escrita como

$$I(x, y, t) = I(x + dx,\ y + dy,\ t + dt), \tag{3.1}$$

onde (x, y) são as coordenadas espaciais do pixel no instante t, e (dx, dy) representam seu deslocamento observado após um intervalo temporal dt.

Para obter uma forma analítica mais útil, expandimos o termo $I(x + dx, y + dy, t + dt)$ em série de Taylor de primeira ordem em torno de (x, y, t), assumindo deslocamentos pequenos:

$$I(x + dx, y + dy, t + dt) \approx I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt. \tag{3.2}$$

Substituindo essa aproximação em (3.1) e cancelando $I(x, y, t)$ em ambos os lados, obtemos, após dividir por dt,

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0. \tag{3.3}$$

Definindo as componentes da velocidade aparente no plano da imagem como $u = \frac{dx}{dt}$ e $v = \frac{dy}{dt}$, chegamos à clássica **equação de restrição do fluxo óptico**:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0. \tag{3.4}$$

Em notação vetorial compacta, escrevemos

$$\nabla I \cdot \mathbf{v} + I_t = 0, \tag{3.5}$$

onde $\nabla I = \left(\frac{\partial I}{\partial x},\ \frac{\partial I}{\partial y}\right)^\top$ é o gradiente espacial da intensidade, $\mathbf{v} = (u, v)^\top$ é o vetor de fluxo óptico e $I_t = \frac{\partial I}{\partial t}$ é o gradiente temporal. Essa relação constitui o ponto de partida para algoritmos de estimação de movimento em sequências de vídeo.

3.2. Solução por Mínimos Quadrados

No contexto do método de Lucas–Kanade, ao considerar uma janela local contendo n pixels ao redor de um ponto de interesse, é possível agrupar as equações de restrição do fluxo óptico em um sistema linear sobre os vetores de intensidade. Para cada pixel i na vizinhança, temos uma equação da forma $I_{xi}u + I_{yi}v = -I_{ti}$. Agrupando todas as n equações em notação matricial, o sistema pode ser representado como:

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \\ I_{tn} \end{bmatrix}, \tag{3.6}$$

ou, de forma compacta, $A\mathbf{v} = \mathbf{b}$, onde A é a matriz de gradientes espaciais, $\mathbf{v} = (u, v)^\top$ é o vetor de fluxo a ser estimado, e $\mathbf{b}$ representa os gradientes temporais com sinal invertido. Como o sistema é geralmente superdeterminado (mais equações do que variáveis), sua solução pode ser obtida por meio do critério dos mínimos quadrados:

$$\mathbf{v} = (A^\top A)^{-1} A^\top \mathbf{b}. \tag{3.7}$$

O termo $(A^\top A)$, que aparece na solução, é conhecido como **matriz de estrutura** e está diretamente relacionado à variação do conteúdo visual dentro da janela considerada. Essa matriz é dada por:

$$M = A^\top A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}. \tag{3.8}$$

A matriz M fornece informações cruciais sobre a textura local da imagem: quando seus autovalores são suficientemente grandes, indica que a região possui variação significativa nas direções x e y, como em cantos ou interseções de arestas. Essas regiões são ideais para estimativa confiável do fluxo óptico, pois garantem bom condicionamento numérico da matriz M e evitam ambiguidade direcional.

Esse princípio motiva critérios de selecionabilidade, como os definidos pelos detectores de cantos de *Harris* e *Shi–Tomasi*, que analisam os autovalores de M para identificar pontos que oferecem estabilidade na estimação de movimento. Em particular, o detector de Shi–Tomasi seleciona regiões onde o menor autovalor de M excede um determinado limiar, garantindo que o problema de fluxo óptico seja bem condicionado e robusto a ruídos e variações locais de intensidade.

4

Estimação Robusta de Transformação

4.1. Método RANSAC

O algoritmo `estimateAffine2D` da biblioteca OpenCV é utilizado para estimar uma transformação afim entre dois conjuntos de pontos correspondentes de forma robusta, mesmo na presença de outliers. Para isso, o método emprega o algoritmo RANSAC (*Random Sample Consensus*)[2], que é amplamente utilizado em visão computacional para lidar com dados ruidosos ou contaminados por correspondências incorretas.

A ideia central do RANSAC é iterar sobre subconjuntos aleatórios mínimos de pontos, ajustando um modelo a cada subconjunto e avaliando quantos dados do conjunto completo são compatíveis com o modelo (isto é, quantos são considerados *inliers*). No caso da estimação afim, o modelo geométrico é definido por uma matriz de transformação $T \in \mathbb{R}^{2\times 3}$, que pode ser completamente determinada por três correspondências ponto-a-ponto.

O algoritmo inicia selecionando aleatoriamente três correspondências do conjunto total $\{(\mathbf{p}_i, \mathbf{p}'_i)\}_{i=1}^{n}$, onde $\mathbf{p}_i$ e $\mathbf{p}'_i$ representam pontos correspondentes nas imagens original e transformada, respectivamente. A partir desses três pares, uma transformação afim T_k é calculada. Em seguida, o algoritmo verifica, para cada ponto restante, se a distância entre o ponto transformado $T_k(\mathbf{p}_i)$ e o correspondente $\mathbf{p}'_i$ é inferior a um limiar predefinido. Os pares que satisfazem essa condição são considerados inliers daquela iteração.

Esse processo é repetido por um número fixo de iterações (ou até que um critério de confiança seja atingido), mantendo-se sempre o conjunto com o maior número de inliers e a respectiva transformação associada. Por fim, a matriz de transformação afim best_T é refinada usando todos os inliers do melhor modelo encontrado, resultando em uma estimativa robusta da geometria entre os dois conjuntos de pontos. O algoritmo retorna tanto a transformação final quanto o subconjunto de inliers identificados.

4.2. Extração de Parâmetros

Considere uma matriz de transformação afim 2×3 definida como:

$$T = \begin{bmatrix} a & b & t_x \\ c & d & t_y \end{bmatrix}, \tag{4.1}$$

onde os coeficientes a, b, c, d estão associados à rotação, escala e cisalhamento, enquanto t_x e t_y representam os componentes de translação ao longo dos eixos x e y, respectivamente. Essa matriz é comumente utilizada para transformar coordenadas de pontos bidimensionais no plano da imagem.

A partir da matriz T, é possível extrair diretamente os parâmetros de movimento. As componentes de translação são obtidas pelas entradas da terceira coluna da matriz:

$$t_x = T[0, 2], \tag{4.2}$$
$$t_y = T[1, 2]. \tag{4.3}$$

Já o ângulo de rotação θ pode ser calculado a partir dos coeficientes da primeira coluna, por meio da função $\arctan 2$, que garante a determinação correta do quadrante:

$$\theta = \arctan 2(T[1, 0], T[0, 0]). \tag{4.4}$$

Esse procedimento permite interpretar geometricamente a transformação afim estimada, separando seus efeitos em termos de translação e rotação, o que é útil em diversas aplicações de estabilização de vídeo, rastreamento de movimento e registro de imagens.

5

Processamento de Sinais e Filtragem

5.1. Filtro de Média Móvel

A função de suavização adotada utiliza um filtro de média móvel, uma técnica simples e eficaz para reduzir flutuações locais em sinais temporais. O filtro é definido pela equação:

$$y[n] = \frac{1}{2R+1} \sum_{k=-R}^{R} x[n+k], \tag{5.1}$$

onde $x[n]$ representa o valor do sinal original no instante n, e $y[n]$ é o valor suavizado correspondente. O parâmetro R define o *raio de suavização*, e o termo $2R+1$ corresponde ao tamanho total da janela centrada em n. Em outras palavras, o valor $y[n]$ é calculado como a média aritmética dos valores de $x[n]$ em uma vizinhança simétrica que se estende de $n-R$ a $n+R$.

Esse tipo de suavização atua como um filtro passa-baixas, atenuando variações de alta frequência e preservando as tendências globais do sinal. É especialmente útil em contextos como estabilização de vídeo, onde pequenas oscilações nos parâmetros estimados entre quadros devem ser eliminadas para produzir movimentos mais suaves e visual

5.2. Resposta em Frequência

A análise da resposta em frequência do filtro de média móvel fornece uma compreensão mais profunda de seu comportamento como sistema passa-baixas. Considerando um filtro com janela simétrica de tamanho $2R + 1$, sua resposta em frequência pode ser expressa como:

$$H(\omega) = \frac{1}{2R+1} \sum_{k=-R}^{R} e^{-j\omega k}, \tag{5.2}$$

onde ω representa a frequência angular normalizada. Essa soma finita de exponenciais complexas é uma forma clássica de série geométrica centrada na origem, cuja solução analítica resulta em:

$$H(\omega) = \frac{\sin\left(\frac{(2R+1)\omega}{2}\right)}{(2R+1)\sin\left(\frac{\omega}{2}\right)}, \tag{5.3}$$

para $\omega \neq 0$. Essa expressão revela que a resposta do filtro se assemelha a uma função do tipo *sinc* normalizada, característica típica de filtros passa-baixas. O numerador controla a forma geral da resposta, enquanto o denominador introduz os nulos periódicos. A principal implicação dessa forma é que o filtro atenua significativamente componentes de alta frequência, o que o torna adequado para suavização de sinais, reduzindo ruídos e oscilações rápidas indesejadas.

5.3. Convolução Discreta

A implementação do filtro de média móvel é realizada por meio de uma convolução discreta entre o sinal de entrada e um kernel de suavização. Em termos gerais, a convolução discreta entre dois sinais $f[n]$ e $g[n]$ é definida como:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m], \tag{5.4}$$

onde $*$ denota a operação de convolução. No caso específico da média móvel, utiliza-se um kernel finito e uniforme definido por $h[k] = \frac{1}{2R+1}$ para $k \in [-R, R]$ e $h[k] = 0$ caso contrário. Isso corresponde a uma janela simétrica centrada na amostra atual, com pesos iguais para cada ponto dentro da janela.

Aplicando essa definição ao sinal de entrada $x[n]$, a saída do filtro suavizado é dada por:

$$y[n] = (x * h)[n] = \sum_{k=-R}^{R} x[n-k] \cdot \frac{1}{2R+1}. \tag{5.5}$$

Essa formulação evidencia que cada valor de saída $y[n]$ é simplesmente a média aritmética dos $2R+1$ valores do sinal $x[n]$ ao redor da posição n, implementada de forma eficiente por convolução. Essa abordagem é amplamente utilizada em processamento digital de sinais por sua simplicidade computacional e sua capacidade de eliminar ruídos de alta frequência sem introduzir grandes distorções de fase ou forma de onda.

5.4. Tratamento de Bordas

Ao aplicar operações de convolução em sinais finitos, é comum utilizar técnicas de *padding* para lidar com as bordas do sinal. Uma das abordagens mais utilizadas é o *padding* do tipo `'edge'`, que consiste em replicar os valores das extremidades do sinal original de forma a estender seu domínio. Esse tipo de extensão evita a introdução de descontinuidades artificiais nas fronteiras e preserva o comportamento local nas regiões limítrofes do sinal.

Matematicamente, o sinal extendido $x_{\text{padded}}[n]$ é definido da seguinte forma:

$$x_{\text{padded}}[n] = \begin{cases} x[0] & \text{se } n < 0, \\ x[n] & \text{se } 0 \leq n < N, \\ x[N-1] & \text{se } n \geq N, \end{cases} \tag{5.6}$$

onde $x[n]$ é o sinal original de comprimento N. Com essa definição, todas as amostras fora do intervalo válido são atribuídas ao valor mais próximo disponível na borda do sinal: o valor inicial $x[0]$ é replicado para os índices negativos, enquanto o valor final $x[N-1]$ é estendido para os índices além do final do vetor. Esse tipo de padding é especialmente útil em filtros como a média móvel, pois minimiza distorções nos primeiros e últimos elementos da saída filtrada.

6

Análise de Trajetória

6.1. Acumulação de Transformações

A trajetória da câmera em uma sequência de vídeo pode ser reconstruída a partir das transformações estimadas entre frames consecutivos. Cada transformação representa o deslocamento da câmera em relação ao quadro anterior, e a posição acumulada ao longo do tempo pode ser obtida somando-se sucessivamente esses deslocamentos. Formalmente, a posição da câmera no instante i é dada por:

$$\text{Trajetória}[i] = \sum_{j=0}^{i} \text{Transformação}[j], \tag{6.1}$$

onde Transformação$[j]$ representa a estimativa de movimento entre os frames j e $j+1$. Essa abordagem assume que os deslocamentos são representados em uma forma aditiva (como vetores de translação ou parâmetros simplificados), o que é comum em estabilização de vídeo baseada em modelos afins ou semelhantes.

Em notação matricial, esse processo pode ser expresso de forma mais compacta como:

$$\mathbf{T}_{\text{traj}} = \text{cumsum}(\mathbf{T}_{\text{frame}}), \tag{6.2}$$

onde $\mathbf{T}_{\text{frame}}$ representa a sequência de transformações estimadas para cada quadro, e a função cumsum$(\cdot)$ denota a soma cumulativa ao longo do tempo. Essa trajetória acumulada é essencial tanto para a análise do movimento da câmera quanto para a aplicação de técnicas de suavização ou correção em estabilização de vídeo.

6.2. Correção de Trajetória

Após a reconstrução da trajetória da câmera, o próximo passo no processo de estabilização consiste em aplicar uma correção que suavize os movimentos indesejados. Para isso, utiliza-se uma versão suavizada da trajetória original, que representa o movimento desejado da câmera sem as oscilações rápidas ou ruídos. Essa suavização pode ser realizada por meio de filtros como a média móvel, resultando em uma trajetória regularizada denotada por $\mathbf{T}_{\text{smooth}}$:

$$\mathbf{T}_{\text{smooth}} = \text{SmoothTrajectory}(\mathbf{T}_{\text{traj}}). \tag{6.3}$$

A correção necessária é então obtida calculando a diferença entre a trajetória suavizada e a trajetória original estimada, o que produz um vetor de deslocamento compensatório:

$$\mathbf{D} = \mathbf{T}_{\text{smooth}} - \mathbf{T}_{\text{traj}}. \tag{6.4}$$

Por fim, essa diferença $\mathbf{D}$ é somada às transformações estimadas entre os frames, gerando um novo conjunto de transformações corrigidas $\mathbf{T}_{\text{corrected}}$ que alinham cada quadro com base na trajetória suavizada:

$$\mathbf{T}_{\text{corrected}} = \mathbf{T}_{\text{frame}} + \mathbf{D}. \tag{6.5}$$

Essa abordagem permite eliminar variações bruscas de movimento, preservando o conteúdo estrutural do vídeo, resultando em uma sequência visualmente mais estável e suave.

7

Análise de Estabilidade e Robustez

7.1. Estabilidade do Filtro de Média Móvel

O filtro de média móvel definido pela equação (5.1) é intrinsecamente estável, uma vez que sua resposta ao impulso satisfaz a condição de somabilidade absoluta. Essa condição é um critério clássico para garantir a estabilidade BIBO (Bounded Input, Bounded Output) em sistemas lineares e invariantes no tempo. Em termos formais, a estabilidade é assegurada quando a soma absoluta da resposta ao impulso é finita:

$$\sum_{n=-\infty}^{\infty} |h[n]| = \sum_{k=-R}^{R} \frac{1}{2R+1} = 1 < \infty, \tag{7.1}$$

onde $h[n]$ representa o kernel do filtro, que neste caso é constante e simétrico em torno da origem. Como todos os coeficientes são positivos e somam exatamente 1, o filtro não amplifica o sinal de entrada, o que reforça sua estabilidade. Além disso, a normalização garante que o ganho em frequência zero (DC) seja unitário, preservando a média do sinal original.

7.2. Análise de Polos e Zeros

A análise do filtro de média móvel no domínio z fornece uma visão mais completa de seu comportamento dinâmico e estabilidade. A função de transferência $H(z)$ do filtro de média móvel com janela simétrica de tamanho $2R+1$ é dada por:

$$H(z) = \frac{1}{2R+1} \sum_{k=-R}^{R} z^{-k} = \frac{1}{2R+1} \cdot \frac{z^{R+1} - z^{-R}}{z-1}. \tag{7.2}$$

Essa expressão representa um filtro FIR (Finite Impulse Response), cujos coeficientes são uniformemente ponderados. A estrutura da equação mostra que o numerador é um polinômio simétrico em z^{-1}, enquanto o denominador é linear. A função de transferência não possui polos fora da origem; todos os seus polos estão localizados em $z = 0$, o que implica que o sistema não apresenta realimentação. Essa característica garante que o filtro seja incondicionalmente estável, ou seja, a resposta do sistema permanecerá limitada para qualquer entrada limitada.

Além disso, a simetria e a finitude da resposta ao impulso reforçam a estabilidade BIBO do sistema, tornando o filtro de média móvel uma escolha segura e eficiente para tarefas de suavização em sinais discretos.

7.3. Estratégias de Robustez Implementadas

O algoritmo foi projetado com diversas estratégias para garantir robustez durante o processo de estimação das transformações afins entre frames consecutivos. Essas estratégias visam lidar com situações comuns em vídeos reais, como perda de pontos de interesse, falhas de correspondência ou exceções durante o processamento.

A primeira verificação implementada consiste na checagem da validade dos pontos de correspondência anteriores, garantindo que existam pelo menos três pares para permitir a estimação de uma transformação afim confiável. Isso é feito com a seguinte condição:

```
if ptsAnterior is not None and len(ptsAnterior) > 2:
```

Caso a função de estimação falhe — seja por ausência de pontos válidos, instabilidade numérica ou erro interno —, o algoritmo utiliza uma matriz identidade como fallback. Essa abordagem assegura que, na ausência de uma estimativa confiável, nenhuma transformação seja aplicada ao frame atual, evitando distorções acumulativas:

```
if iMatriz is None:
    iMatriz = np.array([[1, 0, 0], [0, 1, 0]], dtype=np.float32)
```

Além disso, a chamada para a função `cv.estimateAffine2D` é envolta em um bloco `try/except`, que captura possíveis exceções durante a estimação. Caso ocorra qualquer erro, o algoritmo substitui a matriz inválida por uma matriz identidade, garantindo a continuidade da execução sem interrupções:

```
try:
    iMatriz, _ = cv.estimateAffine2D(ptsAnterior, ptsAtual)
except:
    iMatriz = np.array([[1, 0, 0], [0, 1, 0]], dtype=np.float32)
```

Essas três estratégias combinadas — verificação de pontos válidos, uso de fallback e tratamento de exceções — tornam o algoritmo resiliente a falhas típicas em aplicações práticas de estabilização de vídeo.

7.4. Análise de Propagação de Erro

Em um sistema de estabilização de vídeo, o erro acumulado no resultado final é consequência direta da propagação de incertezas ao longo das diferentes etapas do pipeline. Cada módulo — desde a detecção de pontos de interesse até a aplicação da suavização — contribui para o erro total observado. Esse acúmulo pode ser representado de forma abstrata como:

$$\text{Erro}_{\text{final}} = f(\text{Erro}_{\text{detecção}}, \text{Erro}_{\text{tracking}}, \text{Erro}_{\text{estimação}}, \text{Erro}_{\text{suavização}}), \tag{7.3}$$

onde a função $f(\cdot)$ representa a maneira pela qual os erros individuais de cada etapa interagem e se acumulam ao longo do processo. Essa formulação ressalta a importância de projetar algoritmos robustos em cada estágio para minimizar o impacto no resultado global.

Ao assumir que os erros associados a cada etapa são estatisticamente independentes e seguem distribuições gaussianas com média zero, é possível modelar a variância do erro final como a soma das variâncias individuais. Esse modelo simplificado, mas bastante útil, é expresso por:

$$\sigma^2_{\text{final}} = \sigma^2_{\text{detecção}} + \sigma^2_{\text{tracking}} + \sigma^2_{\text{estimação}} + \sigma^2_{\text{suavização}}. \tag{7.4}$$

Essa abordagem fornece uma estimativa quantitativa do erro total e destaca a importância de otimizar cada componente do sistema, já que a redução de erro em qualquer uma das etapas contribui diretamente para a melhoria da precisão global da estabilização.

8

Complexidade Computacional e Otimização

8.1. Detecção de Features

A complexidade computacional de algoritmos de rastreamento baseados em pontos de interesse depende fortemente das etapas de detecção e estimação de movimento. No caso da detecção de features, o método de Shi–Tomasi é amplamente utilizado por sua robustez e precisão na seleção de pontos bem condicionados. A complexidade desse algoritmo pode ser expressa como:

$$O_{\text{detecção}} = O(W \times H \times B^2), \tag{8.1}$$

onde $W \times H$ representa a resolução da imagem, e B é o tamanho do bloco local (janela) utilizado na análise da matriz de autocorrelação. Essa dependência quadrática em relação ao tamanho do bloco deve-se ao cálculo dos autovalores da matriz de estrutura para cada região da imagem, o que envolve operações sobre uma vizinhança bidimensional.

Para a estimação de movimento, o algoritmo de fluxo óptico piramidal de Lucas–Kanade é comumente empregado, permitindo o rastreamento eficiente de pontos em diferentes escalas. A complexidade associada a essa etapa é dada por:

$$O_{\text{flow}} = O(N \times L \times W^2), \tag{8.2}$$

em que N é o número total de pontos rastreados, L é o número de níveis da pirâmide de resolução (normalmente decrescentes em potência de dois), e W é o tamanho da janela de cálculo em cada nível. A dependência quadrática em W decorre da necessidade de resolver sistemas lineares locais baseados na janela ao redor de cada ponto em cada escala da pirâmide.

Ambas as expressões destacam o impacto direto da resolução da imagem, do número de pontos e do tamanho das janelas nos custos computacionais dos algoritmos, sendo fatores cruciais para decisões de otimização em implementações de estabilização de vídeo em tempo real.

8.2. Optical Flow

A análise de complexidade computacional é essencial para entender o custo das etapas envolvidas em um pipeline de estabilização de vídeo. O algoritmo de fluxo óptico piramidal de Lucas–Kanade é amplamente utilizado para o rastreamento de pontos em vídeos devido à sua eficiência e precisão em múltiplas escalas. A complexidade desse método pode ser expressa como:

$$O_{\text{flow}} = O(N \times L \times W^2), \tag{8.3}$$

onde N representa o número de pontos rastreados, L é o número de níveis da pirâmide de resoluções, e W corresponde ao tamanho da janela usada para estimar o deslocamento local. A complexidade cresce linearmente com o número de pontos e níveis da pirâmide, mas quadraticamente com o tamanho da janela, já que cada ponto é processado considerando uma vizinhança bidimensional de tamanho $W \times W$.

Em seguida, durante o processo de estabilização, aplica-se uma etapa de suavização sobre os parâmetros de movimento ao longo do tempo. Quando essa suavização é feita por meio de um filtro de média móvel, a complexidade é proporcional ao número de quadros e ao tamanho da janela de suavização. Essa relação pode ser expressa por:

$$O_{\text{smooth}} = O(F \times R \times P), \tag{8.4}$$

onde F é o número total de frames no vídeo, R representa o raio da janela da média móvel (ou seja, metade do tamanho total da janela), e P é o número de parâmetros suavizados por frame (por exemplo, translação em x, y e rotação). Embora essa etapa seja computacionalmente mais simples do que a estimação do fluxo óptico, ela desempenha papel fundamental na regularização da trajetória estimada e na supressão de ruídos de alta frequência, contribuindo diretamente para a qualidade visual da estabilização final.

8.3. Complexidade Total

A complexidade computacional total por quadro em um sistema de estabilização de vídeo pode ser decomposta como a soma das complexidades individuais de cada etapa do pipeline. Essas etapas geralmente incluem a detecção de pontos de interesse, o cálculo do fluxo óptico, a estimação da transformação geométrica e a aplicação da transformação à imagem (warping). Assim, a complexidade total por frame é expressa como:

$$O_{\text{total}} = O_{\text{detecção}} + O_{\text{flow}} + O_{\text{estimação}} + O_{\text{warp}}, \tag{8.5}$$

onde $O_{\text{detecção}}$ representa o custo da detecção de features (por exemplo, com o método de Shi–Tomasi), O_{flow} refere-se à estimação do movimento com fluxo óptico, $O_{\text{estimação}}$ corresponde ao cálculo da transformação global (como uma transformação afim via RANSAC), e O_{warp} diz respeito à aplicação da transformação aos pixels da imagem. Cada uma dessas etapas contribui para o custo computacional total por quadro, sendo fundamentais para avaliar a viabilidade de implementação do sistema em tempo real ou em ambientes com restrições de recursos.

9

Parâmetros e Calibração

9.1. Raio de Suavização

O parâmetro R, que define o raio da janela no filtro de média móvel, desempenha um papel fundamental no controle do equilíbrio entre suavidade e responsividade da trajetória suavizada. Um valor maior de R tende a produzir trajetórias mais suaves, pois o filtro considera uma vizinhança temporal mais ampla, atenuando flutuações de alta frequência. No entanto, esse aumento de suavidade vem à custa da responsividade, introduzindo um maior atraso na resposta do sistema em relação a mudanças reais no movimento da câmera.

Essa relação caracteriza um clássico trade-off entre estabilidade visual e fidelidade temporal, que pode ser formalizado por meio de uma função de custo. O valor ótimo de R pode ser obtido minimizando-se uma combinação ponderada entre a variância do movimento residual (indicador de oscilação não suavizada) e o atraso introduzido pela suavização:

$$R_{\text{ótimo}} = \arg\min_{R} \left(\text{Var}(\text{movimento residual}) + \lambda \cdot \text{Lag}(\text{resposta}) \right), \tag{9.1}$$

onde λ é um parâmetro de regularização que controla a penalização do atraso. Ajustar adequadamente esse valor permite adaptar a suavização ao contexto da aplicação — por exemplo, priorizando resposta rápida em vídeos interativos ou maior suavidade em vídeos cinematográficos.

9.2. Parâmetros de Detecção

Os parâmetros utilizados na etapa de detecção de pontos de interesse exercem influência direta na qualidade da estimação de movimento e, consequentemente, no desempenho global da estabilização de vídeo. A escolha adequada desses parâmetros afeta tanto a densidade quanto a confiabilidade dos pontos rastreados ao longo dos quadros.

Entre os principais parâmetros, destaca-se o `maxCorners`, que define o número máximo de pontos de interesse a serem detectados. Valores maiores aumentam a cobertura espacial, mas também elevam o custo computacional e o risco de incluir pontos instáveis:

$$\text{maxCorners} : \text{Número máximo de pontos.} \tag{9.2}$$

Outro parâmetro importante é o `qualityLevel`, que estabelece um limiar relativo de qualidade para os pontos detectados. Ele atua como um filtro que descarta pontos com baixa resposta de canto, permitindo controlar a confiabilidade dos pontos selecionados:

$$\text{qualityLevel} : \text{Limiar de qualidade.} \tag{9.3}$$

Por fim, o parâmetro `minDistance` especifica a distância mínima permitida entre pontos vizinhos, evitando que múltiplos pontos sejam detectados em regiões muito próximas da imagem. Esse controle ajuda a distribuir os pontos de forma mais uniforme e a reduzir a redundância:

$$\text{minDistance} : \text{Distância mínima entre pontos.} \tag{9.4}$$

O ajuste cuidadoso desses parâmetros é essencial para equilibrar densidade, precisão e desempenho computacional, e deve ser adaptado ao conteúdo visual e às exigências da aplicação.

9.3. Otimização Multi-objetivo

A escolha adequada dos parâmetros envolvidos no processo de estabilização pode ser formulada como um problema de otimização multicritério, no qual se busca um equilíbrio entre diferentes objetivos conflitantes. A função de custo total é composta por uma combinação ponderada de critérios, cada um refletindo um aspecto importante da qualidade e eficiência do sistema. A formulação geral do problema é dada por:

$$\min_{\boldsymbol{\theta}} \left[w_1 \cdot f_1(\boldsymbol{\theta}) + w_2 \cdot f_2(\boldsymbol{\theta}) + w_3 \cdot f_3(\boldsymbol{\theta}) \right], \tag{9.5}$$

em que $\boldsymbol{\theta}$ representa o vetor de parâmetros a serem otimizados, e os pesos w_1, w_2 e w_3 determinam a importância relativa de cada termo. Os objetivos individuais são definidos da seguinte forma:

$f_1(\boldsymbol{\theta})$: Variância do movimento residual, que quantifica a suavidade da trajetória estabilizada; (9.6)

$f_2(\boldsymbol{\theta})$: Lag temporal introduzido pelo filtro de suavização, relacionado à responsividade do sistema; (9.7)

$f_3(\boldsymbol{\theta})$: Custo computacional associado à configuração dos parâmetros, relevante para aplicações em tempo real. (9.8)

Essa formulação permite ajustar o sistema de estabilização de acordo com os requisitos específicos da aplicação, priorizando suavidade, responsividade ou desempenho computacional conforme necessário. A solução ótima $\boldsymbol{\theta}^*$ corresponderá ao ponto de compromisso ideal entre esses critérios.

9.4. Invariância à Translação

Um aspecto importante da robustez do método de Lucas–Kanade é sua invariância à translação pura. Esse comportamento é formalizado no seguinte teorema:

Teorema 1 (Invariância à Translação)**.** *Se a imagem transformada é dada por* $I'(x, y) = I(x - t_x, y - t_y)$*, então o algoritmo detecta corretamente o vetor de translação* (t_x, t_y)*.*

Demonstração. Quando o movimento entre dois quadros consiste exclusivamente em uma translação, o campo de movimento é constante em toda a imagem. Nesse caso, o modelo de fluxo óptico assume a forma:

$$\frac{\partial I}{\partial x} \cdot t_x + \frac{\partial I}{\partial y} \cdot t_y + \frac{\partial I}{\partial t} = 0, \tag{9.9}$$

onde os termos $\frac{\partial I}{\partial x}$ e $\frac{\partial I}{\partial y}$ representam os gradientes espaciais, e $\frac{\partial I}{\partial t}$ é o gradiente temporal da intensidade da imagem. Como essa equação é linear em relação aos parâmetros de translação t_x e t_y, ela pode ser resolvida diretamente e com precisão pelo método de Lucas–Kanade, que emprega uma abordagem de mínimos quadrados sobre uma janela local. Portanto, em situações de translação pura, o algoritmo recupera exatamente o vetor de deslocamento aplicado. □

9.5. Invariância à Rotação

Em muitas aplicações de visão computacional e processamento geométrico, especialmente na modelagem de transformações locais, é comum assumir que os ângulos de rotação são suficientemente pequenos para permitir aproximações lineares. Nessa condição, as funções trigonométricas seno e cosseno podem ser simplificadas pelas suas expansões de Taylor de primeira ordem. Para ângulos θ próximos de zero (medidos em radianos), valem as aproximações:

$$\sin\theta \approx \theta, \tag{9.10}$$

$$\cos\theta \approx 1. \tag{9.11}$$

Essas aproximações são amplamente utilizadas para simplificar expressões analíticas em estimativas de movimento, decomposições de transformações e análises de estabilidade, permitindo tratar rotações pequenas como deslocamentos quase lineares. Embora simplificadas, tais aproximações apresentam excelente precisão para ângulos menores que aproximadamente 10°, o que as torna bastante úteis em contextos como estabilização de vídeo, onde as transformações entre frames consecutivos tendem a ser sutis.

9.6. Suposições do Modelo

O algoritmo de estabilização baseado em fluxo óptico e transformações afins parte de um conjunto de hipóteses que, quando satisfeitas, garantem o funcionamento adequado da estimativa de movimento. A primeira suposição é a existência de um **movimento global dominante**, o que significa que o deslocamento da câmera é o principal componente de variação entre os frames, enquanto os movimentos internos da cena são desprezíveis. Em segundo lugar, assume-se um **modelo afim** para a transformação geométrica, o que limita a capacidade do sistema de capturar deformações não-lineares, como projeções complexas ou efeitos de paralaxe.

Adicionalmente, o algoritmo presume **iluminação constante** entre os quadros consecutivos, uma condição necessária para a validade da equação de fluxo óptico, que assume que a intensidade dos pixels permanece inalterada ao longo do tempo. Por fim, considera-se que o **movimento entre frames seja pequeno**, uma vez que a linearização da equação diferencial do fluxo óptico é válida apenas para deslocamentos sutis.

No entanto, em cenários práticos, essas suposições podem ser violadas, resultando em falhas na estimação. Entre os principais **casos de falha**, destacam-se situações de **movimento rápido**, onde o módulo do vetor de movimento excede o limiar admissível, isto é, $\|\mathbf{v}\| > \text{threshold}$. Também ocorrem problemas em **mudanças abruptas de iluminação**, que invalidam a relação $\frac{\partial I}{\partial t} = -\nabla I \cdot \mathbf{v}$ assumida no modelo.

Além disso, **oclusões** — quando objetos desaparecem ou surgem entre os quadros — comprometem a correspondência entre pontos, afetando diretamente a estimativa do movimento. Outro caso crítico ocorre em **superfícies com baixa textura**, onde o gradiente espacial da imagem é próximo de zero, ou seja, $\|\nabla I\| \approx 0$, o que inviabiliza a solução confiável do sistema linear utilizado pelo método de Lucas–Kanade.

Essas limitações evidenciam a importância de pré-processamentos adequados e do uso de estratégias robustas, como a rejeição de outliers e o uso de múltiplos pontos distribuídos na imagem, para garantir maior estabilidade e precisão na estimativa de movimento.

9.7. Transformação Perspectiva

A transformação perspectiva, também conhecida como homografia, é um modelo geométrico amplamente utilizado em visão computacional para descrever como os pontos de um plano são mapeados entre duas imagens diferentes, geralmente capturadas sob diferentes pontos de vista. Esse tipo de transformação é particularmente eficaz em contextos onde a câmera sofre movimento projetivo, como rotação, translação e variações de perspectiva.

Matematicamente, a homografia é representada por uma matriz 3×3 que atua sobre coordenadas homogêneas. A relação entre um ponto original (x, y) e seu correspondente transformado (x', y') é dada por:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{9.12}$$

onde os coeficientes h_{ij} compõem a matriz de homografia, com um total de oito graus de liberdade (o último elemento é fixado em 1 para garantir uma escala arbitrária). Após a multiplicação, os valores resultantes devem ser normalizados dividindo-se as duas primeiras coordenadas pelo fator de homogeneização (terceiro elemento do vetor resultante). Esse modelo é essencial em tarefas como mosaico de imagens, estabilização de vídeos em planos inclinados, retificação e registro entre imagens com perspectiva diferente.

Referências

[1] Vedant Gaur. «Lucas-Kanade Optical Flow Machine Learning Implementations». Em: *Journal of Student Research* 11 (ago. de 2022). DOI: 10.47611/jsrhs.v11i3.2957.

[2] Ziv Yaniv. «Random Sample Consensus (RANSAC) Algorithm, A Generic Implementation». Em: *The Insight Journal* (out. de 2010). DOI: 10.54294/ia6mzx.