

# KLT Revisited: Spatio-Temporal Affine Regression for Feature Tracking

KLT Revisitado: Regressão Afim Espaço-Temporal para Rastreamento de Características

Nigel Joseph Bandeira Dias<sup>1\*</sup>, Gustavo Teodoro Laureano<sup>1</sup>, Ronaldo Martins Da Costa<sup>1</sup>

**Abstract:** Feature association is an essential step for vision-based localization methods. These methods rely on feature matching to estimate relative motion between consecutive frames using projective geometry. Regardless of the advances in feature association, most existing methods still rely on pairwise feature matching approaches and ignore the rich temporal context in image sequences. In this paper, we revisit the well-known Kanade-Lucas-Tomasi (KLT) feature tracker algorithm and propose a differentiable tracker model. The Proposed method is a fully convolutional neural network that learns spatio-temporal features to track keypoints across videos. Experimental results show that the proposed method outperforms the KLT method, especially in challenging environments.

**Keywords:** feature — tracking — spatio-temporal — deep neural networks

**Resumo:** A correspondência de pontos de interesse (características) entre imagens é uma etapa fundamental em algoritmos de localização visual. Nesses métodos, o movimento relativo entre os quadros de um vídeo é estimado a partir da associação dessas características, utilizando princípios da geometria projetiva. Apesar dos avanços recentes, a maioria das abordagens realiza essa correspondência considerando apenas informações espaciais entre pares de imagens, negligenciando a informação temporal disponível na sequência de imagens. Neste trabalho, apresentamos um modelo neural para o rastreamento de características inspirado no algoritmo de Kanade-Lucas-Tomasi (KLT). Este trabalho apresenta uma arquitetura de rede neural convolucional (CNN) projetada para aprender a extrair características espaço-temporais para rastreamento de cantos/pontos de interesse em vídeos. Os resultados experimentais demonstram que o método proposto supera o desempenho do KLT tradicional, especialmente em cenários visualmente desafiadores que sofrem com problemas de textura.

**Palavras-Chave:** características — rastreamento — espaço-temporal — redes neurais profundas

<sup>1</sup> Instituto de Informática, Universidade Federal de Goiás (UFG), Brazil

\*Corresponding author: nigeldias@hotmail.com

DOI: <http://dx.doi.org/10.22456/2175-2745.150903> • Received: 14/10/2025 • Accepted: 27/10/2025

CC BY-NC-ND 4.0 - This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

## 1. Introduction

Vision-based localization methods have their basis in geometric modeling of the world in which most algorithms attempt to estimate these geometric models from perceived data [1]. The performance of these algorithms relies heavily on the feature association step to estimate relative motion between consecutive frames using projective geometry [2].

Despite significant advances in feature association, especially with deep neural networks, most of the existing methods focus on pairwise feature matching approaches. In general, these methods rely on sparse interest points that are matched using high-dimensional representations that encode their local visual appearance. Such representations often come at the cost of high computational requirements and increased implementation complexity [3, 4]. Likewise, as they rely on

pairwise matching, they typically overlook the rich temporal context in the image sequence.

An alternative to the feature matching approach, which requires extracting features from all candidate images and then performing computationally expensive matching operations, is to detect features once and then track them across subsequent images [5, 6]. This tracking-based method offers significant computational advantages by avoiding the need to repeatedly extract and match features in every frame, making it particularly suitable for real-time applications where processing efficiency is crucial.

Deep neural networks have driven recent advances in point tracking; however, existing methods [7, 8] primarily focus on long-range video. Their designs are robust to occlusions by recovering points after they reappear, but this robustness comes

at a high computational cost. Furthermore, these trackers often neglect geometric consistency, a critical property for applications like structure-from-motion (SfM).

When there is a large visual overlap and limited appearance change between consecutive frames, a window can be tracked by optimizing a matching criterion over a limited range of possible transformations [9]. This approach is more suitable for applications such as Visual Odometry (VO) and Visual SLAM (V-SLAM) where the expected amount of motion and appearance deformation between adjacent frames is expected to be small.

In this paper, we revisit the well-known Kanade-Lucas-Tomasi (KLT) [5, 9] feature tracker and rebuild this classic approach using deep neural concepts for spatiotemporal and geometric learning, such as Inflated 3D convolution [10] and Spatial Transformer Networks [11].

Preliminary results demonstrated that the proposed method outperforms the KLT method, providing more precise and consistent results, especially in challenging environments. The results suggest that learned spatio-temporal features can provide stronger and more discriminative signals than the raw image gradients.

This paper is organized as follows. Section II reviews the Kanade-Lucas-Tomasi (KLT) method and its formulation. Section III details our proposed method, followed by the experimental results and discussion in Section IV. We conclude the paper in Section V.

## 2. Background

### 2.1 The brightness constancy assumption

An image sequence can be represented by a function  $I(x, y, t)$ , where  $(x, y)$  are the spatial coordinates and  $t$  is the time variable. Although camera and object motion can cause complex intensity changes, consecutive frames in a sequence are typically highly correlated. This temporal redundancy forms the basis for motion estimation, which assumes that the frame at time  $t + 1$  can be approximated by displacing the pixels in the frame at time  $t$ . The movement of each pixel can be described by a displacement vector  $\mathbf{d} = [d_x \ d_y]$ :

$$I(x, y, t + 1) = I(x - d_x, y - d_y, t) \quad (1)$$

The assumption of brightness constancy, formulated in equation 1, is frequently not satisfied even under theoretically ideal conditions, such as a static scene with uniform illumination. For instance, noise can change a pixel's value and similarity to its neighbors can cause confusion, making it difficult or impossible to track the pixel between frames. A better approach is to enrich the motion model for a window by defining a set of possible displacement functions. An affine motion field is a practical choice because of its simplicity and capacity of describing complex displacements. Therefore, the displacement of a point  $\mathbf{x}$  between two images  $I$  and  $J$  is expressed by the following equation:

$$J(A\mathbf{x} + \mathbf{d}) = I(\mathbf{x}) \quad (2)$$

where  $A = \mathbf{1} + \mathbf{D}$ ,  $\mathbf{1}$  is a  $2 \times 2$  identity matrix and  $\mathbf{D}$  is a deformation matrix:

$$\mathbf{D} = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \quad (3)$$

For a given window centered at  $\mathbf{x}$  in image  $I$ , the goal of tracking is to find its corresponding region in image  $J$ . This is achieved by estimating the six parameters of an affine motion model, which in turn define a deformation matrix  $\mathbf{D}$  and a displacement vector  $\mathbf{d}$ .

### 2.2 The aperture problem

The quality of the estimated parameters is highly sensitive to several factors: the size of the feature window, the texture content within that window, and the magnitude of motion between frames. The choice of window size presents a fundamental trade-off between estimation stability and model validity. A smaller window makes the matrix  $\mathbf{D}$  harder to estimate because it contains less motion variation, which reduces reliability. Still, small windows are usually better for tracking since they're more likely to include features at similar depths.

Therefore, an affine motion is employed to compare features between the initial and current frames to assess their quality, whereas a pure translation model is preferable for tracking purposes:

$$J(\mathbf{x} + \mathbf{d}) = I(\mathbf{x}) \quad (4)$$

where the deformation matrix  $\mathbf{D}$  is assumed to be zero. The pure translation model provides more stable and reliable results than affine transformations when the inter-frame camera motion is small. On the other hand, an affine model is necessary to compare distant frames, as is required when determining dissimilarity.

The problem of determining the motion parameters can be formulated as finding  $\mathbf{A}$  and  $\mathbf{d}$  that minimize the dissimilarity:

$$\varepsilon = \iint_W [J(A\mathbf{x} + \mathbf{d}) - I(\mathbf{x})]^2 w(\mathbf{x}) d\mathbf{x} \quad (5)$$

where  $W$  denotes the given feature window and  $w(\mathbf{x})$  is a weighting function.

The formulations described above outline the Kanade-Lucas-Tomasi (KLT) feature tracker. This method was first proposed [5] as a computationally efficient solution for image registration. A subsequent paper [12] improved the technique by introducing a selection criterion for identifying features that are suitable for tracking. Then, another contribution [9] added a verification stage to ensure tracking consistency. Later, a pyramidal implementation was developed [13] to handle large motions, resolving the trade-off between local accuracy and robustness associated with the integration window's size.

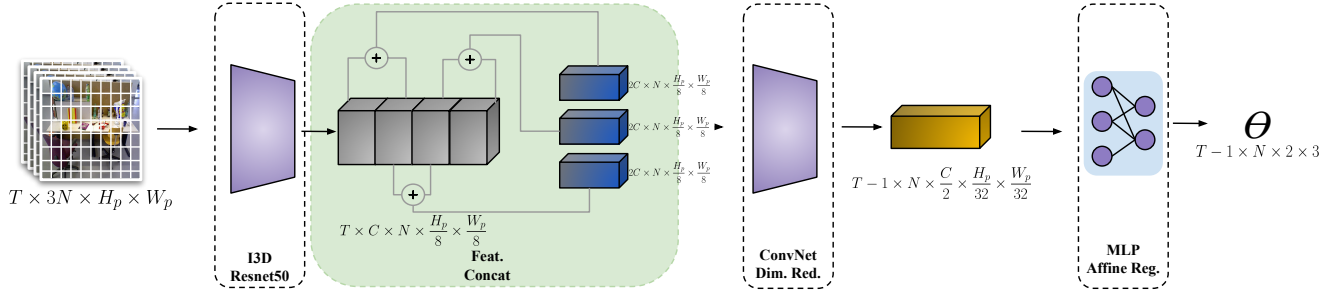


**Figure 1. Overview of the Proposed Tracking Architecture.** The model takes an RGB clip as input. The clip is first decomposed into a grid of patches, which are then fed into a spatio-temporal feature extractor. To capture motion, the extracted features from consecutive frames are concatenated and passed through a convolutional layer for dimensionality reduction. Finally, the resulting feature map is processed by a Multi-Layer Perceptron (MLP) to regress the affine transformation matrix for each patch.

### 3. Proposed Differentiable Tracker model

We propose a novel, differentiable tracking model inspired by the classic Kanade-Lucas-Tomasi (KLT) [5, 9] algorithm. The proposed model is a fully convolutional neural network that learns to predict the affine transformation parameters between patches from consecutive frames, enabling precise tracking of interest points. Its architecture consists of a patch-based spatio-temporal feature extractor and a regression block that uses these features to predict the affine parameters of each patch. The proposed method is illustrated in Figure 1.

#### 3.1 Patch-Based Spatio-Temporal encoder

The model's backbone uses an Inflated 3D ConvNet (I3D) [10] architecture for feature extraction. This architecture builds upon successful 2D ConvNets, such as ResNet, by inflating their filters and pooling kernels into 3D. This inflation process allows the model to learn seamless spatio-temporal features directly from video while retaining the proven designs of ImageNet [14] architectures. Consequently, a key advantage is the ability to use pre-trained 2D weights as a valuable initialization, which significantly accelerates the training process.

Our primary goal is tracking specific interest points, a task focused on local detail rather than the global scene analysis common in video understanding [15]. To focus on the local nature of interest point tracking, we propose a patch-based method.

The encoder takes as input a set of non-overlapping patches  $\mathbf{X} \in \mathbb{R}^{N \times T \times C \times H_p \times W_p}$ , where  $N$  is the number of patches,  $T$  is the temporal dimension, and  $(H_p, W_p)$  are the patch's spatial dimensions. It outputs a dense feature map  $\mathbf{F} \in \mathbb{R}^{N \times T \times D \times H_f \times W_f}$ , where  $D$  represents the feature dimension. Each patch is processed independently, enabling parallel computation. This localized analysis also ensures the extraction of coherent spatio-temporal features from regions likely to share a similar depth.

The I3D ConvNet architecture is designed to extract semantic-level spatiotemporal features, making it well-suited for tasks such as action recognition and event understanding. Given our focus on establishing correspondence at the patch

level, we use early-to-mid feature maps rather than the deepest layers. These intermediate representations retain higher spatial resolution and preserve structural details, which is essential for capturing local spatial and short-term temporal patterns such as edges, corners, textures, and localized motion dynamics.

#### 3.2 Transformation Regressor

The transformation Regressor is a lightweight regression head that predicts the parameters of the  $3 \times 2$  affine transformation matrix similar to the one implemented in the Spatial Transformer Network [16].

The regressor head is implemented as a small multilayer perceptron (MLP), consisting of two linear layers with a non-linear activation function in between. To capture temporal dynamics, the input is prepared in several steps. First, feature maps from consecutive frames,  $F^t$  and  $F^{t+1}$ , are concatenated along their channel dimension. This creates a motion-focused tensor  $\mathbf{M} \in \mathbb{R}^{N \times T-1 \times 2D \times H_f \times W_f}$ . This combined representation then passes through a convolutional layer for dimensionality reduction and feature refinement. Finally, the resulting features are fed into the MLP head, allowing the model to effectively learn the geometric relationships between frames.

The regressor head learns a mapping function  $f$  that estimates the affine transformation parameters,  $\theta$ , for each patch  $p$  based on its features in consecutive frames  $f(F_p^t, F_p^{t+1})$ . The output  $\theta$  represents the 6 parameters of an affine transformation that maps pixel coordinates from frame  $t$  to frame  $t+1$ :

$$\begin{pmatrix} x_i^{t+1} \\ y_i^{t+1} \\ 1 \end{pmatrix} = \theta_p \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} d_{xx} & d_{xy} & d_x \\ d_{yx} & d_{yy} & d_y \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \quad (6)$$

where the two vectors of the first column represent the deformation matrix  $\mathbf{D}$  and the last column represents the displacement vector  $\mathbf{d}$ , as described in 2.

### 3.3 Loss Functions

The transformation regressor is trained under the pure translation constraint described in Equation 1, by supervising the translation component with a displacement loss. This loss measures the discrepancy between the predicted displacements  $\hat{\mathbf{d}}_{p,t} \in \mathbb{R}^2$  and the ground-truth displacements  $\mathbf{d}_{p,t}^{\text{gt}} \in \mathbb{R}^2$ , as detailed in Section 3.4. To improve robustness to outliers, the displacement loss is computed using the  $L_1$  distance:

$$\mathcal{L}_{\text{disp}} = \frac{1}{N_{\text{valid}}} \sum_{b,p,t} m_{bpt} \cdot \left\| \hat{\mathbf{d}}_{bpt} - \mathbf{d}_{bpt}^{\text{gt}} \right\|_1 \quad (7)$$

where indices  $b$ ,  $p$  and  $t$  denotes batch, patch and time, respectively. The variable  $m_{bpt} \in \{0, 1\}$  is a binary mask indicating valid training supervision—specifically, whether patch  $p$  in frame  $t$  has a valid displacement vector. This ensures that only valid patches contribute to the loss, while  $N_{\text{valid}}$  normalizes by the total number of valid elements. Note that raw pixel displacements are scale-dependent. To stabilize training and ensure the loss remains comparable across patch sizes and image resolutions, both predicted and ground-truth displacements are normalized within each patch relative to its size.

We found that regressing the point trajectories in an optical-flow-like manner, rather than directly regressing the points themselves, improves model convergence. The predicted displacements are estimated between consecutive frames  $f_t \mapsto f_{t+1}$ , whereas the ground-truth displacements are generated between the reference frame and all subsequent frames within the clip  $f_0 \mapsto f_t$ . Before computing the loss, the predicted affine matrices are composed along the time dimension to match the temporal structure of the ground truth. This strategy enhances the accuracy of long-term tracking.

To encourage temporal consistency and smoothness, we introduce a temporal smoothness loss that penalizes large variations in the predicted displacements between consecutive frames:

$$\mathcal{L}_{\text{smooth}} = \frac{1}{N_{\text{valid}}} \sum_{b,p,t} m_{bpt}^{\text{diff}} \cdot \left\| \hat{\mathbf{d}}_{b,p,t+1} - \hat{\mathbf{d}}_{b,p,t} \right\|_2^2 \quad (8)$$

where  $m_{bpt}^{\text{diff}} = m_{b,p,t} \wedge m_{b,p,t+1}$  ensures that both time steps have valid supervision. This regularization term discourages unrealistic temporal jumps in the predicted motion, promoting smooth, physically plausible trajectories for point tracking.

To solve the full affine motion problem described in Equation 5, we combine the displacement regression term  $\mathcal{L}_{\text{disp}}$  with a photometric term  $\mathcal{L}_{\text{photo}}$ , computed by warping patches using the predicted affine matrices. The photometric loss integrates structural similarity (SSIM) [17] and the  $L_1$  distance to evaluate visual consistency between warped patches and the reference frame:

$$\mathcal{L}_{\text{photo}} = \frac{1}{N_{\text{valid}}} \sum_{b,p,t} m_{b,p,t} \left[ \alpha_{\text{ssim}} (1 - \text{SSIM}(W(I_{b,p,t}, \theta_{b,p,t}), I_{b,p,0})) + (1 - \alpha_{\text{ssim}}) \|W(I_{b,p,t}, \theta_{b,p,t}) - I_{b,p,0}\|_1 \right] \quad (9)$$

where  $W(\cdot, \theta)$  denotes the warping function parameterized by the transformation  $\theta$ ,  $I_{b,p,0}$  is the reference frame, and  $I_{b,p,t}$  is the current frame. The SSIM component captures structural information by evaluating perceptual similarity with robustness to illumination changes, while the  $L_1$  term enforces strict pixel-level accuracy. The  $\alpha_{\text{ssim}}$  parameter balances these two complementary measures, with a higher value prioritizing structural correctness over pixel-wise fidelity.

The final loss  $\mathcal{L}$  is then a linear combination of all losses with  $\mathcal{L}_{\text{disp}}$  as the main loss:

$$\mathcal{L} = \mathcal{L}_{\text{disp}} + \lambda_{\text{smooth}} \cdot \mathcal{L}_{\text{smooth}} + \lambda_{\text{photo}} \cdot \mathcal{L}_{\text{photo}} \quad (10)$$

where  $\lambda_{\text{smooth}}$ ,  $\lambda_{\text{photo}}$  and  $\alpha_{\text{SSIM}}$  hyperparameters that control the relative contribution of each term. The combined loss function addresses multiple aspects of point tracking: accuracy (displacement), temporal consistency (smoothness) and visual coherence (photometric).

### 3.4 Dataset generation

The available datasets for feature correspondence [18] are primarily designed for feature matching approaches, which aim to establish correspondences between images with large deformations. Such datasets are not suitable for the proposed method. Similarly, existing datasets for point tracking in videos [19, 20] were not created for feature correspondence evaluation. These datasets generally aim to track arbitrary points in videos, with ground-truth data often generated by following objects that move independently of the camera. Consequently, the resulting data violate the rigid-body motion assumption, which is fundamental to Visual Odometry (VO) and Visual SLAM (V-SLAM) tasks.

Therefore, we created our own dataset, built upon the TUM RGB-D benchmark [21]. This dataset was chosen because it provides high-quality ground-truth camera trajectories and depth maps, enabling both local displacement evaluation and global odometry metric computation. In addition, it offers a wide variety of scenes and camera motions well suited to our application. The dataset consists of 47 sequences, each comprising  $640 \times 480$  resolution images recorded at 30Hz in both office environments and an industrial hall.

To avoid temporal leakage, we adopt a video-level separation for data splitting: 70% of sequences are allocated for training, 15% for validation, and 15% for testing. The original video sequences are segmented into fixed-length, non-overlapping clips (i.e., sets of consecutive frames). No clip in the training set shares frames from the same original sequence as those in the validation or test sets, ensuring

**Table 1.** Key parameters and values considered at training time.

| Hyper-parameters   | Values |
|--------------------|--------|
| learning rate      | 6e-4   |
| Batch size         | 12     |
| Clip length        | 4      |
| $\lambda_{smooth}$ | 0.013  |
| $\lambda_{photo}$  | 0.013  |
| $\alpha_{ssim}$    | 0.75   |

the model does not encounter nearly identical content across splits. Within each split, we employ a clip-sampling strategy: clips within the same sequence may overlap, but no overlap is permitted across different splits.

Ground-truth per-patch displacements and validity masks are generated offline at the original image resolution using SuperPoint [22] keypoints and SuperGlue [3] matches, in order to accelerate training. For each clip, keypoints are extracted from the reference frame and matched with all subsequent frames within the clip  $f_0 \mapsto f_i$ . To remove incorrect correspondences, matches are filtered using the fundamental matrix, ensuring they are geometrically plausible rather than merely visually similar.

### 3.5 Implementation Details

**Data Preprocessing.** During training, we use clips with a temporal length of  $T = 4$ . Input frames are first resized to  $256 \times 256$  pixels, from which we extract non-overlapping  $32 \times 32$  patches. For each clip, a maximum of 1000 features are extracted from the initial frame. The displacement vectors from the resulting matches are then associated with their corresponding patches based on feature location.

**Data Augmentation.** We apply a series of data augmentations to improve robustness, including random adjustments to brightness and contrast, as well as the addition of Gaussian noise, speckle noise, and motion blur.

**Architecture.** Our feature encoder is based on a ResNet-50 architecture [23]. As stated previously, we remove the final residual blocks to extract low- to mid-level features.

**Training Details.** The model was trained for 120,000 steps using the Adam optimizer with a learning rate of  $6 \times 10^{-4}$ . We used a batch size of 12 and trained on a single NVIDIA GeForce RTX 4090 GPU. The model's memory consumption scales primarily with the clip length,  $T$ .

The key hyperparameters, presented in Table 1, were optimized via a Bayesian search algorithm [24] to maximize validation accuracy. The search results revealed that  $\lambda_{smooth}$  is the most influential parameter, exhibiting a strong positive correlation with performance. This emphasizes the importance of enforcing temporal smoothness for the model. Conversely, the weights for the photometric loss,  $\lambda_{photo}$  and  $\alpha_{ssim}$ , demonstrated moderate and low importance, respectively. This suggests that while the visual feedback from the photometric loss  $\mathcal{L}_{photo}$  is beneficial, the model relies more significantly on the direct supervision from the displacement loss  $\mathcal{L}_{disp}$ .

## 4. Experiments and results

The proposed method, detailed in Section 3, is compared to the Kanade-Lucas-Tomasi (KLT) Feature Tracker implementation from [13]. Which is a pyramidal implementation of the classic KLT, designed to address the aperture problem described in Section 2. To ensure a direct and fair comparison, we configure the KLT baseline to use a window size equal to the patch size of our method.

For evaluation, we defined a set of metrics to quantify the geometric accuracy of the feature trackers. These metrics aim to measure how well the tracked points obey geometric constraints imposed by camera motion, such as epipolar geometry [1], which is useful to evaluate their utility for downstream tasks such as VO, V-SLAM and 3D reconstruction. All experiments were conducted on the Freiburg 3 sequences, which was selected because its color and IR images are already undistorted, meaning the distortion parameters are all zero [21]. The sequences are divided into non-overlapping clips of four frames each ( $T = 4$ ). Within each clip, SuperPoint features [22] are extracted from the first frame and tracked to the last, generating a set of correspondences. This resulting set of correspondences is then used for further evaluation. The implementation of all methods detailed below is based on the OpenCV library [25].

### 4.1 Pose-derived metrics

Pose-derived metrics are task-oriented, as they directly measure how the quality of feature tracking translates into pose estimation performance. The evaluation process is as follows: for each clip, the generated set of feature correspondences is used to compute the Essential matrix that best fits the data. This is done within a RANSAC framework to ensure robustness to outliers. The Essential matrix is then decomposed to yield an estimated relative pose ( $\mathbf{R}_{est}, \mathbf{t}_{est}$ ), which is compared to the ground-truth relative pose ( $\mathbf{R}_{gt}, \mathbf{t}_{gt}$ ) derived from the dataset's trajectory.

The rotation error, which quantifies the angular drift caused by correspondence inaccuracies, is computed through the residual rotation  $\mathbf{R}_{rel} = \mathbf{R}_{gt}^T \cdot \mathbf{R}_{est}$ :

$$\theta_{error} = \cos^{-1} \left( \frac{\text{trace}(\mathbf{R}_{rel}) - 1}{2} \right) \quad (11)$$

Figure 2 presents the resulting statistics on this rotational discrepancy for each method. The violin plots show a clear performance advantage for the proposed method over KLT. Our method exhibits both a lower median error and an error distribution that is more tightly clustered around this median, indicating that our tracker provides more precise and consistent results.

The second metric is the translation error, defined as the Euclidean distance between the estimated and ground-truth translation vectors. It is worth noting that the translation vector obtained from Essential matrix decomposition has an inherent scale ambiguity; it can only be recovered up to an

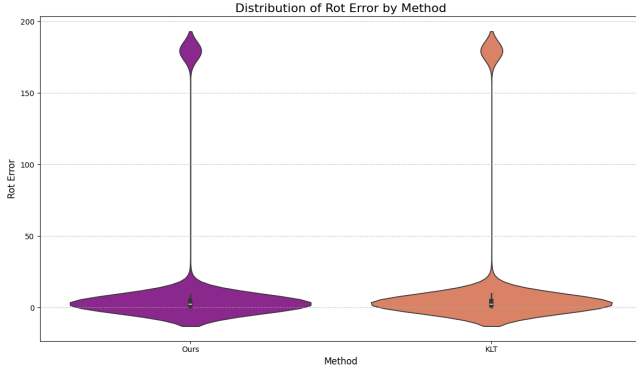


**Figure 2.** Violin plot of the relative rotation error in degrees

unknown scale factor. Therefore, before computing the Euclidean distance, the estimated translation vector must be aligned with the ground-truth scale. This is achieved by first scaling the estimate:

$$\mathbf{t}_{est}^{scaled} = \mathbf{t}_{est} \cdot \|\mathbf{t}_{gt}\| \quad (12)$$

the final translation error is then calculated as:

$$trans_{error} = \|\mathbf{t}_{gt} - \mathbf{t}_{est}^{scaled}\| \quad (13)$$

A statistical comparison of the translation error for each method is presented in Figure 3. The superior performance of the proposed method extends to translation estimation, where it again achieves greater precision and consistency compared to KLT.

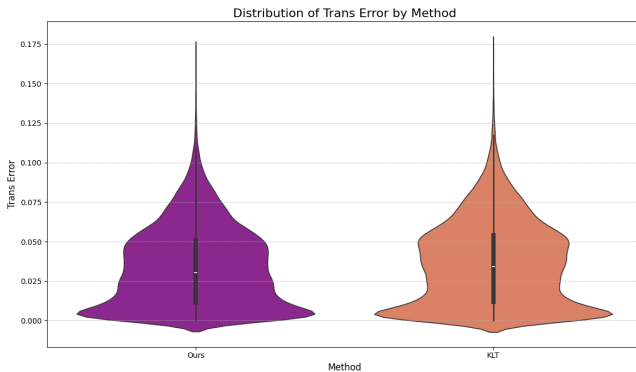


**Figure 3.** Violin plot of the relative translation error in meters

## 4.2 Geometric accuracy metric

The geometric accuracy metric aims to measure the robustness and reliability of a feature tracker by verifying if its feature correspondences are geometrically consistent with the ground-truth camera motion. Unlike the pose-derived metrics, this evaluation is completely decoupled from the performance of a RANSAC-based pose estimator. Instead, it purely assesses the quality of the raw matches produced by the tracker.

For each feature correspondence we calculate its Sampson distance, which is a first-order approximation of the geometric error between a point and its corresponding epipolar line [1]. Using the ground truth relative pose, we compute the ground truth essential matrix using the geometric relationship:

$$\mathbf{E}_{gt} = [\mathbf{t}_{gt}]_{\times} \cdot \mathbf{R}_{gt} \quad (14)$$

where  $[\mathbf{t}_{gt}]_{\times}$  represents the skew-symmetric matrix operator that converts a 3D vector to its corresponding cross-product matrix. The essential matrix is transformed to the fundamental matrix through camera calibration:

$$\mathbf{F}_{gt} = \mathbf{K}^{-T} \cdot \mathbf{E}_{gt} \cdot \mathbf{K}^{-1}. \quad (15)$$

Given the fundamental matrix the Sampson distance can be computed as follows:

$$d = \frac{(\mathbf{x}'^T \mathbf{F} \mathbf{x})^2}{(\mathbf{F} \mathbf{x})_1^2 + (\mathbf{F} \mathbf{x})_2^2 + (\mathbf{F}^T \mathbf{x}')_1^2 + (\mathbf{F}^T \mathbf{x}')_2^2} \quad (16)$$

The resulting statistics for this metric are presented for each method in Figure 4. It is important to note that for the Sampson distance metric, we use the median error per clip instead of the mean, as it is more robust to outliers. This choice is reflected in the plot, which again indicates our method's superiority. The KLT distribution's longer tail suggests that while both methods can fail, KLT's failures result in significantly larger errors compared to our approach.

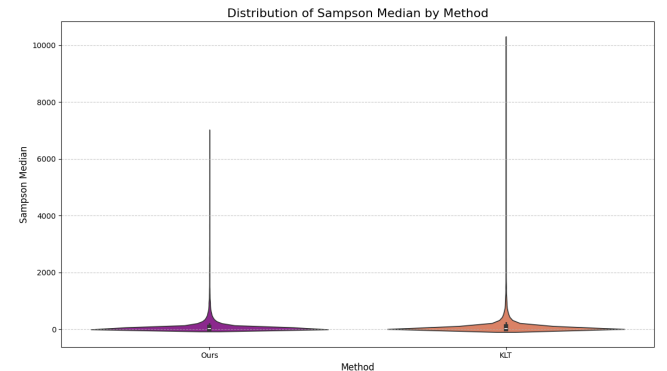


**Figure 4.** Violin plot of the Sampson distance in pixels

**Table 2.** Statistics of the sampson distance in pixels

| Method | Mean            | Median         | Std             |
|--------|-----------------|----------------|-----------------|
| KLT    | 122.2110        | 27.0485        | 356.9791        |
| Ours   | <b>104.5471</b> | <b>23.9995</b> | <b>281.4559</b> |

## 4.3 Discussion

Table 3 summarizes the overall performance of the compared methods, corroborating the findings from the violin plot analysis. As a whole, our proposed method outperforms the KLT

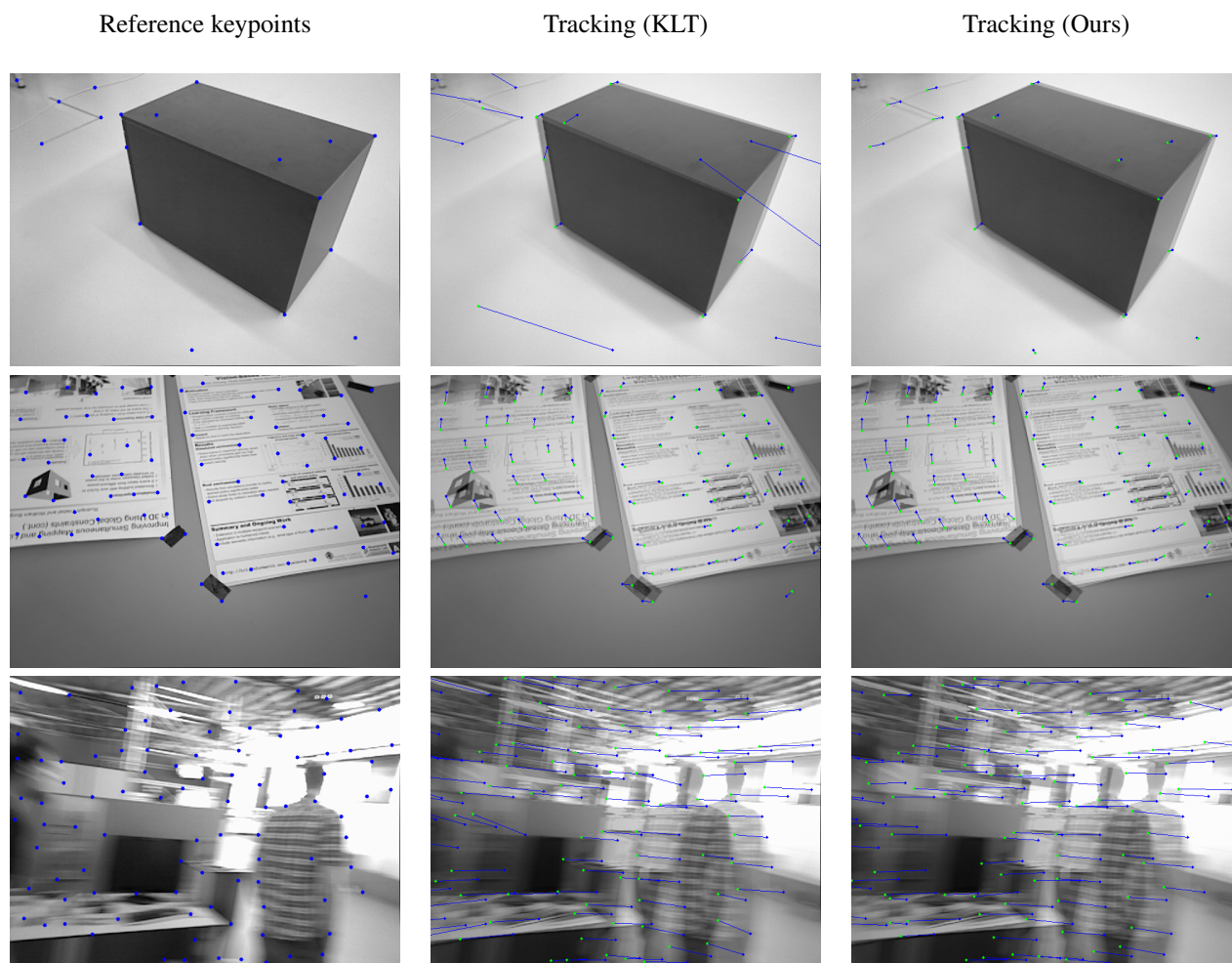


**Figure 5. Qualitative Results on the TUM Dataset.** A comparison between our proposed tracker and the KLT baseline across three challenging sequences.

baseline. While both methods are capable of failing, our approach proves to be significantly more consistent.

These results suggest that the learned features provide stronger and more discriminative signals than the raw image gradients used by KLT. This advantage is particularly evident in low-texture or ambiguous regions, where our method can reduce mismatches, as illustrated by the examples in Figures 6, 7, and 5.

**Table 3.** Overall performance summary

| Method | Trans err<br>(m) | Rot err<br>(deg) | Samp dist<br>(px) |
|--------|------------------|------------------|-------------------|
| KLT    | 0.0360           | 17.1145          | 27.0485           |
| Ours   | <b>0.0336</b>    | <b>16.8980</b>   | <b>23.9995</b>    |

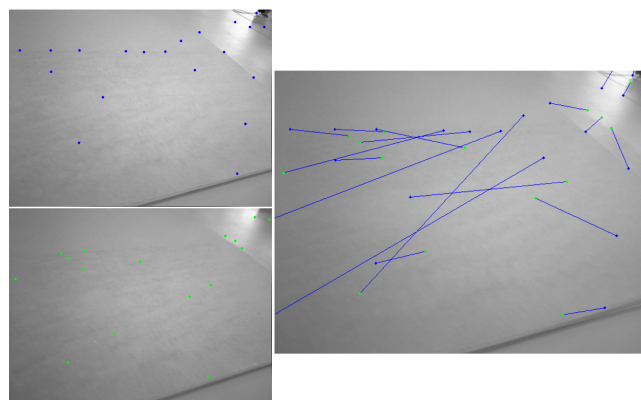


**Figure 6.** Qualitative Evaluation of the KLT Algorithm in an Unstructured and Textureless Environment

## 5. Conclusion

In this paper, we proposed a differentiable feature tracker inspired by the KLT algorithm. Our method combines a spatio-temporal feature extractor and a transformation regressor head

that predicts an affine motion model between patches from consecutive frames. The experimental results show that the proposed method can be more robust than KLT, especially in unstructured and low-texture environments. We argue that

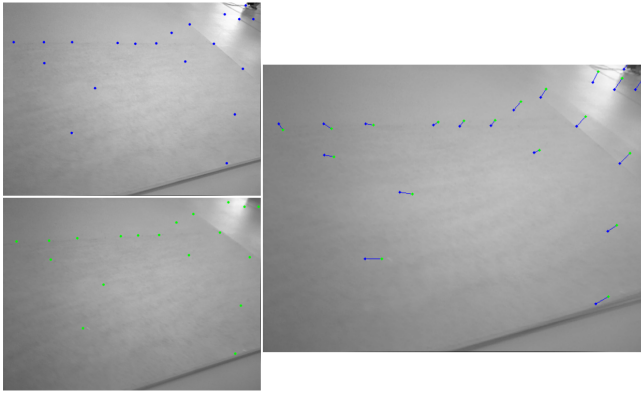


**Figure 7.** Qualitative Evaluation of the proposed method in an Unstructured and Textureless Environment

this robustness is related to the fact that learned features tend to be more resilient to illumination changes, blur, and small geometric deformations. They can also be less sensitive to the aperture problem by providing stronger and more discriminative signals than raw image gradients, which reduces mismatches in low-texture or ambiguous regions.

Although the proposed method has demonstrated promising results, further improvements are needed. It is important to note that KLT is still much faster and more lightweight than our approach. Likewise, the proposed method should be evaluated on other datasets to confirm its robustness against the KLT method.

Future work might include extending the method to other datasets, evaluating different backbones and clips with longer lengths, integrating the tracker into VO and V-SLAM pipelines, and performing an extensive comparison with other matching approaches.

## Acknowledgements

This work was supported by CEIA (Centro de Excelência em Inteligência Artificial, Artificial Intelligence Center of Excellence).

## Author Contributions

All authors (Dias, N.J.B, Laureano, G.T, and Da Costa, R.M.) were involved in the conceptualization, methodology, formal analysis, and visualization of this study. Dias, N.J.B prepared the original draft, which was reviewed and edited by all authors. Laureano, G.T, and Da Costa, R.M. provided supervision and performed the validation of the results.

## References

- [1] HARTLEY, R.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. 2. ed. [S.l.]: Cambridge University Press, 2004.
- [2] FRAUNDORFER, F.; SCARAMUZZA, D. Visual odometry : Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics and Automation Magazine*, v. 19, n. 2, p. 78–90, 2012.
- [3] SARLIN, P.-E. et al. Superglue: Learning feature matching with graph neural networks. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. p. 4937–4946.
- [4] POTJE, G. et al. Xfeat: Accelerated features for lightweight image matching. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2024. p. 2682–2691.
- [5] LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981. (IJCAI'81), p. 674–679.
- [6] SAND, P.; TELLER, S. Particle video: Long-range motion estimation using point trajectories. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. [S.l.: s.n.], 2006. v. 2, p. 2195–2202.
- [7] HARLEY, A. W.; FANG, Z.; FRAGKIADAKI, K. Particle video revisited: Tracking through occlusions using point trajectories. In: *ECCV*. [S.l.: s.n.], 2022.
- [8] DOERSCH, C. et al. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2023. p. 10061–10072.
- [9] SHI, J.; TOMASI. Good features to track. In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 1994. p. 593–600.
- [10] CARREIRA, J.; ZISSERMAN, A. Quo vadis, action recognition? a new model and the kinetics dataset. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017. p. 4724–4733.
- [11] JADERBERG, M. et al. Spatial transformer networks. In: *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*. Cambridge, MA, USA: MIT Press, 2015. (NIPS'15), p. 2017–2025.
- [12] TOMASI, C. Detection and tracking of point features. In: . [s.n.], 1991. Disponível em: <https://api.semanticscholar.org/CorpusID:238434334>.
- [13] BOUGUET, J.-Y. Pyramidal implementation of the lucas kanade feature tracker. In: . [s.n.], 1999. Disponível em: <https://api.semanticscholar.org/CorpusID:9350588>.
- [14] DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2009. p. 248–255.
- [15] WANG, X.; JABRI, A.; EFROS, A. A. *Learning Correspondence from the Cycle-Consistency of Time*. 2019. Disponível em: <https://arxiv.org/abs/1903.07593>.

- [16] JADERBERG, M. et al. *Spatial Transformer Networks*. 2016. Disponível em: <https://arxiv.org/abs/1506.02025>.
- [17] WANG, Z.; SIMONCELLI, E.; BOVIK, A. Multiscale structural similarity for image quality assessment. In: *The Thrity-Seventh Asilomar Conference on Signals, Systems and Computers, 2003*. [S.l.: s.n.], 2003. v. 2, p. 1398–1402 Vol.2.
- [18] LI, Z.; SNAVELY, N. Megadepth: Learning single-view depth prediction from internet photos. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 2041–2050.
- [19] DOERSCH, C. et al. *TAP-Vid: A Benchmark for Tracking Any Point in a Video*. 2023. Disponível em: <https://arxiv.org/abs/2211.03726>.
- [20] GREFF, K. et al. Kubric: a scalable dataset generator. 2022.
- [21] STURM, J. et al. A benchmark for the evaluation of rgb-d slam systems. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. [S.l.: s.n.], 2012.
- [22] DETONE, D.; MALISIEWICZ, T.; RABINOVICH, A. *SuperPoint: Self-Supervised Interest Point Detection and Description*. 2018. Disponível em: <https://arxiv.org/abs/1712.07629>.
- [23] HE, K. et al. *Deep Residual Learning for Image Recognition*. 2015. Disponível em: <https://arxiv.org/abs/1512.03385>.
- [24] DEWANCKER, I.; MCCOURT, M.; CLARK, S. Bayesian optimization primer. URL <https://app.sigopt.com/static/pdf/SigOpt-Bayesian-Optimization-Primer.pdf>, 2015.
- [25] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.