Video stabilization algorithm for feld robots in uneven terrain

University of Trento

Dario Tortorici

Accademic Year 2023/2024

Contents

1 Introduction

This project concerns a video stabilisation algorithm created for field robots that navigate uneven terrains. The algorithm is detailed in the paper *Video Stabilization Algorithm for Field Robots in Uneven Terrain* by Abhijeet Ravankar, Arpit Rawankar, and Ankit A. Ravankar, published in 2023 here. The purpose of this algorithm is to stabilise camera motion by using feature points extracted from frames and applying geometric transformations to compensate for any motion irregularities. This algorithm is primarily used in vineyard robots, assuming favorable illumination conditions during operation.

1.1 Flowchart

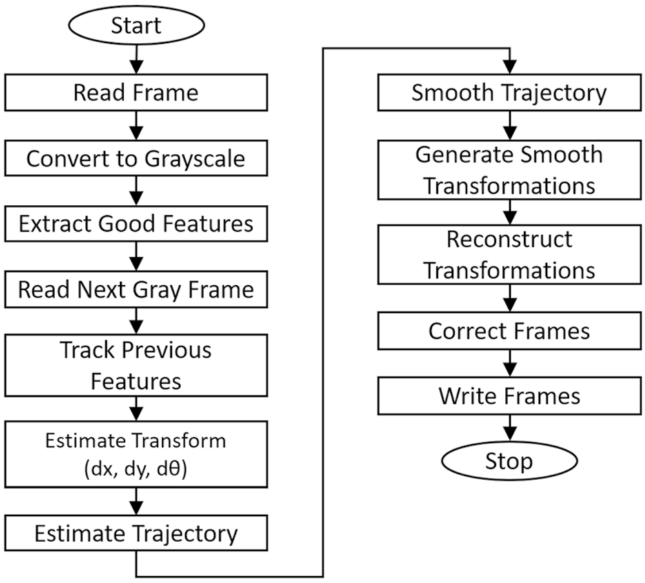

Figure 1: Flowchart of the proposed algorithm

1. **Input Processing:** The algorithm starts by converting input frames to grayscale, for computational efficiency without compromising algorithm accuracy.

2. **Feature Extraction and Tracking:**
   - The process of feature extraction depends on the Shi-Tomasi method, which involves identifying salient points within frames.
   - The Lucas-Kanade method facilitates feature tracking, ensuring consistent point tracking across frames.
   - For a reliable tracking, the paper consider a 3x3 patch around each point and assume shared motion.

3. **Optical Flow Equation:** The optical flow equation is represented as:

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

   where $f_x u + f_y v + f_t = 0$ describes the equation governing optical flow.
   Here, $f_x = \frac{\partial f}{\partial x}$, $f_y = \frac{\partial f}{\partial y}$, $u = \frac{dx}{\partial dt}$, and $v = \frac{dy}{dt}$.

4. **Optical Flow Solution:** The algorithm solves the optical flow equation utilizing the least squares method, ensuring accurate motion estimation.

5. **Cumulative Trajectory Calculation:** By summing the transformations across frames, a cumulative trajectory is computed, providing an overall view of the camera's motion.

6. **Smoothing and Convolution:** The algorithm uses a moving average method to calculate a smooth trajectory, which reduces erratic variations in the camera's path.

7. **Correction Parameters Estimation:** The algorithm calculates the difference between the smooth and cumulative trajectories and derives correction parameters to achieve accurate stabilization.

8. **Final Correction Transformation:** The final correction transformation matrix M_{sc} is constructed as follows:
$$M_{sc} = \begin{bmatrix} \cos(\theta_{\text{correct}}) & -\sin(\theta_{\text{correct}}) & dx_{\text{correct}} \\ \sin(\theta_{\text{correct}}) & \cos(\theta_{\text{correct}}) & dy_{\text{correct}} \end{bmatrix}$$
The correction parameters are encapsulated in this matrix, which facilitates stabilization by applying suitable transformations to the frames.

2 Feature Extraction Using Shi-Tomasi Method

It is a cornerstone of our algorithm's feature extraction phase, although the paper allows for some degree of freedom in choosing the algorithm for this stage. The Shi-Tomasi method is crucial in identifying standout points within an image, particularly corners or regions exhibiting significant intensity variations.

2.1 Corner Detection

The algorithm begins by examining small image patches. For each pixel, it calculates a score based on local intensity variations in both the x and y directions. This scoring mechanism allows for the identification of potential corner points.

2.2 Non-Maximum Suppression

After scoring corners, a non-maximum suppression process is applied. This involves analyzing the local neighborhood around each pixel to keep only the pixels with the highest scores within a specific window. Weaker corner candidates are discarded, preserving only the most prominent corners.

2.3 Quality Thresholding

After non-maximum suppression, the algorithm applies a quality threshold. Corners with scores below this threshold are considered lower in quality and therefore removed. This filtering guarantees that only corners meeting a certain distinctiveness or quality criterion are processed further.

2.4 Maximum Distance Parameter

In addition, the algorithm includes a parameter for the maximum distance between corners. This parameter excludes points that are too close to each other, in order to prevent confusion between similar corners and ensure accurate detection. These measures prevent complications in the subsequent phase of optical flow detection, ensuring a fairer and more accurate representation of the frame's characteristics.

3 Optical Flow Equation and Stabilization Process

After the discovery of Shi-Tomasi indicators within the first frame, the procedure proceeds to track these indicators over subsequent frames, utilising the Lucas-Kanade technique. This strategy relies on the assumption that neighbouring pixels exhibit similar motion, and uses a 3x3 region surrounding each Shi-Tomasi indicator to create a coherent motion pattern between them. This technique allows the algorithm to map the path of these features across frames.

3.1 Optical Flow Equation

The optical flow equation is expressed as:

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

where $f_x u + f_y v + f_t = 0$ describes the equation governing optical flow.

Here, $f_x = \frac{\partial f}{\partial x}$, $f_y = \frac{\partial f}{\partial y}$, $u = \frac{dx}{\partial dt}$, and $v = \frac{dy}{dt}$.

Figure 2: Optical flow example of three Shi-Tomasi points

3.2 Optical Flow Solution

The algorithm resolves the optical flow equation using the least squares method, ensuring accurate estimation of motion dynamics.

4 Smoothing Trajectory with Moving Average Method

The optical flow trajectory is smoothed using a moving average method. This involves mathematical components such as window size (W), padding, convolution, and parameter estimation to ensure good stabilisation.

4.1 Window Size (W)

The window size (W) defines the span over which the smoothing effect is applied to discrete segments within the vineyard. Mathematically, the moving average calculation within a window of size W, denoted as N, is represented as:

$$\text{Smoothed Value}(t) = \frac{1}{W} \sum_{i=t}^{t+W-1} \text{Original Value}(i)$$

Where t represents the current time index, W is the window size, and Original Value(i) denotes the original data at time index i.

4.2 Padding and Convolution

Padding augments the data sequence to ensure uninterrupted computation of the moving average. The convolution operation within the window size (W) N is mathematically expressed as:

$$\text{Convolution}(t) = \frac{1}{W} \sum_{i=t}^{t+W-1} \text{Original Value}(i) \times \text{Kernel}(i)$$

Where Kernel(i) represents the weighting factor at time index i within the window.

4.3 Trajectory Refinement

Utilizing the moving average method, the algorithm refines the trajectory to minimize abrupt variations. Mathematically, this controlled smoothing is crucial in stabilizing the optical flow trajectory:

$$\text{Smoothed Trajectory}(t) = \frac{1}{W} \sum_{i=t}^{t+W-1} \text{Optical Flow}(i)$$

Where Optical Flow(i) represents the optical flow data at time index i.

4.4 Parameter Estimation

Analysis of the contrast between the smoothed trajectory and the cumulative trajectory aids in deriving accurate correction parameters:

$$\text{Correction Parameters} = \text{Cumulative Trajectory} - \text{Smoothed Trajectory}$$

This estimation ensures precise alignment and adjustment of the camera's motion for effective stabilization.

4.5 Optimized Window Size Selection

Prior to deployment, the robot traverses the vineyard to determine the optimal window size (W) based on the vineyard layout. In the paper, the researchers empirically set it to 50, 75 and 100. In my test, the window size for the selected video was empirically set to 100, as this produces a better performance.

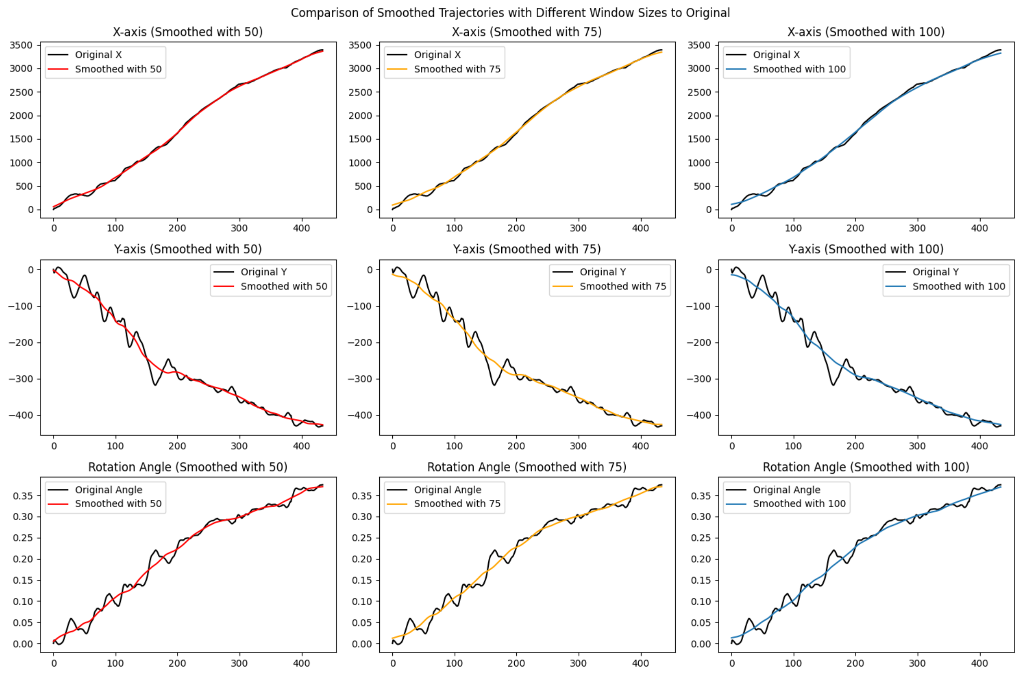

Figure 3: Comparison of Original and Smoothed Trajectories with different window sizes

4.6 Final Stabilization Transformation

The ultimate step involves constructing the final correction transformation matrix M_{sc}, integrating the calculated correction parameters to stabilize the frames:

$$M_{sc} = \begin{bmatrix} \cos(\theta_{\text{correct}}) & -\sin(\theta_{\text{correct}}) & dx_{\text{correct}} \\ \sin(\theta_{\text{correct}}) & \cos(\theta_{\text{correct}}) & dy_{\text{correct}} \end{bmatrix}$$

Where θ_{correct} represents the corrected angle and $dx_{\text{correct}}, dy_{\text{correct}}$ denote the corrected displacement in x and y directions, respectively.

5 Stabilization and Artifact Mitigation

When frames are stabilized, artifacts such as black borders around corners may appear. To address these issues, previous research has proposed methods like inpainting. However, due to their computational expense, the authors of this research deliberately avoided using them for practical reasons.

Instead, a pragmatic approach is used to enlarge the corrected frame by a certain percentage, denoted as δ %, from the center of the frame. The value of δ is determined empirically beforehand. For instance, in the experimental setups conducted in vineyards, a δ value of 5% was utilized. In my test video, the δ was set to 10% because the footage was more shaky.

This enlargement from the center effectively reduces black border artifacts without the computational overhead of more complex artifact removal techniques.

6 Conclusions

6.1 Pros

This method is distinguished by its exclusive reliance on camera information for video stabilization, which sets it apart from alternative algorithms that incorporate data from inertial measurement units (IMUs) or encoders. By solely utilizing camera data, the proposed method achieves stabilization, altough with certain constraints related to rapid environmental changes and the nature of the robot's movement patterns within the vineyard.

6.2 Cons

The proposed method has limitations, particularly with sudden illumination changes and abrupt occlusions. These factors challenge the stabilization process, affecting the algorithm's ability to maintain consistent video quality in the presence of rapid lighting variations or unexpected obstructions.
The algorithm's effectiveness decreases when faced with significant robot movements, particularly large rotations. To mitigate this issue, the robot was programmed to navigate mainly along the center of the vineyard lanes. As a result, instances requiring significant rotations to avoid oncoming obstacles were infrequent, limiting the need for extensive stabilization due to substantial motion. In addition, my implementation of the algorithm cannot be run live because it is slightly slower than the time required.