Landslide Detection Using Digital Image Correlation and Vegetation Segmentation

Abstract

This paper presents a novel approach for landslide detection that integrates Digital Image Correlation (DIC) with Random Forest based vegetation segmentation using positioned camera systems. The methodology combines real-time motion detection through FFT-based cross-correlation with vegetation masking to effectively filter out vegetation-induced noise which enhance landslide/change detection accuracy in mountainous terrain. The integration of multi-modal feature extraction with morphological post-processing achieves robust landslide identification with high precision and computational efficiency. Experimental results show significant improvements in detection accuracy, particularly in complex terrain with mixed vegetation coverage while computational efficiency enables real-time monitoring applications.

Introduction

Landslides represent one of the most destructive natural hazards worldwide, causing substantial economic losses and endangering human lives. Traditional landslide monitoring systems rely on geological surveys, ground-based sensors, or satellite imagery, which suffer from limitations in temporal resolution, spatial coverage, or accessibility in remote mountainous regions. Digital Image Correlation (DIC) has emerged as a powerful technique for measuring surface displacements by comparing sequential images of the same terrain.

This research presents a practical method for landslide detection that can be easily implemented using fixed cameras. The approach enables digital tracking of landslide motion through Digital Image Correlation (DIC) applied to sequential images. In addition, the study introduces a vegetation filtering approach to minimize noise and interference from vegetation movement, thereby improving the reliability and accuracy of DIC-based displacement measurements.The integration of DIC with vegetation segmentation offers a promising solution to enhance landslide detection accuracy while maintaining the computational efficiency required for real-time monitoring applications.

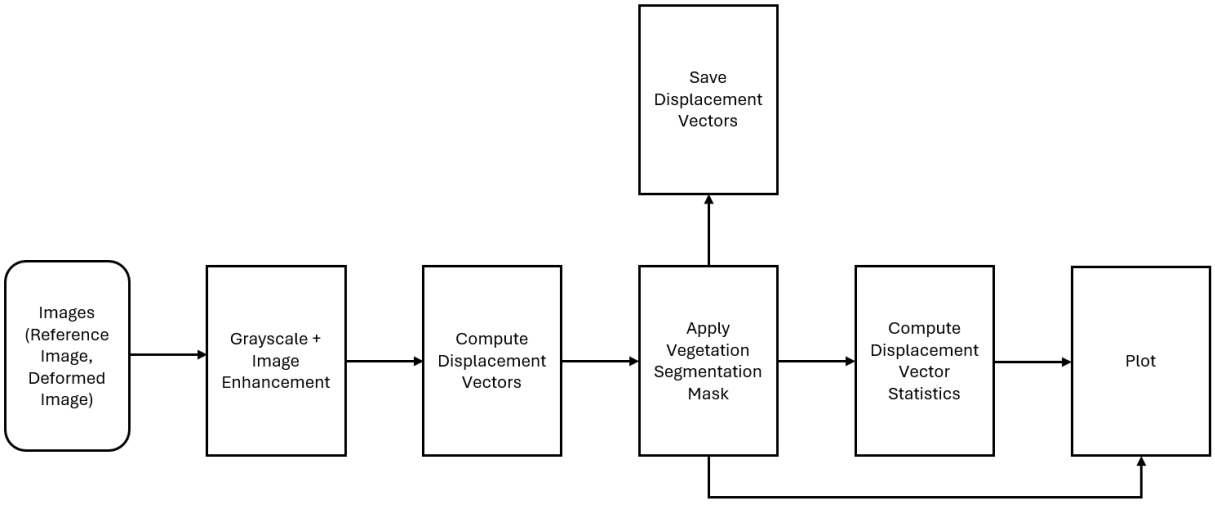
Key contributions include:

1. **Integrated Processing Pipeline:** A sequential approach that first applies DIC to estimate displacement vectors and then uses vegetation segmentation to exclude vegetation regions, resulting in more reliable landslide motion detection.
2. **DIC Implementation:** Application of FFT-based DIC with subpixel refinement, adapted for landslide detection. The implementation retains extensibility for incorporating subset-shape functions to further improve motion-tracking precision with higher-resolution imagery.
3. **Feature-Based Vegetation Classification:** Development of a comprehensive feature set (45-dimensional) for accurate vegetation classification, improving the segmentation process and minimizing its impact on displacement estimation.

Methodology

High Level Overview:

This proposed landslide detection system employs a two part pipeline system that analyzes camera images for terrain displacement and vegetation coverage and then applies the vegetation segmentation mask on the detected vegetation areas to remove those displacement vectors from the image. Finally the statistics of displacement vectors is computed and the image with the statistics is plotted.
The code is provided in the Github repository: [DasAnup356/Landslide-Detection]



Digital Image Correlation Algorithm for Motion Detection

The input are two images, the reference image and the deformed image.
In ideal case, the two images should be closest to each other, which means both should have very small temporal difference, (the condition is just that the change in the image should be small).
In the case, where due to some reason the camera has moved or shifted, some image registration algorithm like ORB or SIFT can be used to get the homography to align the two images and then crop the two images to the aligned regions before putting them as input.

Both the Images should be of the same size, although the algorithm in theory should be able to work on two images of different dimensions but to make the code simple and elegant, the usage is restricted to input images of only the same size.

Reference Image Deformed Image

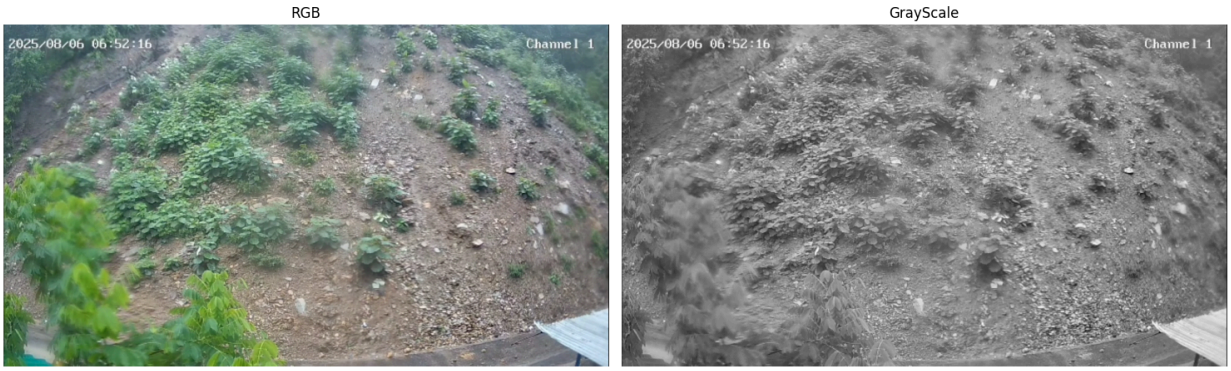
[The Figure shows Input Images: Reference Image and Deformed Image;
The Images are taken by a fixed Camera at 1920*1080 px resolution, at an interval of 10 sec;]

[Image: On aligning one top of another it can be seen that the marked regions show some movement whereas the rest of the image seems almost stationary;

Image Preprocessing and Enhancement

Grayscale:
A grayscale is applied on the two images because the DIC algorithm works on only the intensity of each pixel as input so the RGB image is required to convert to a grayscale image.

(PIL library was used to read and grayscale images, while using matlplotlib.imread to read images didn't work for some internal reasons in the library (unknown))



[Grayscaled Image]

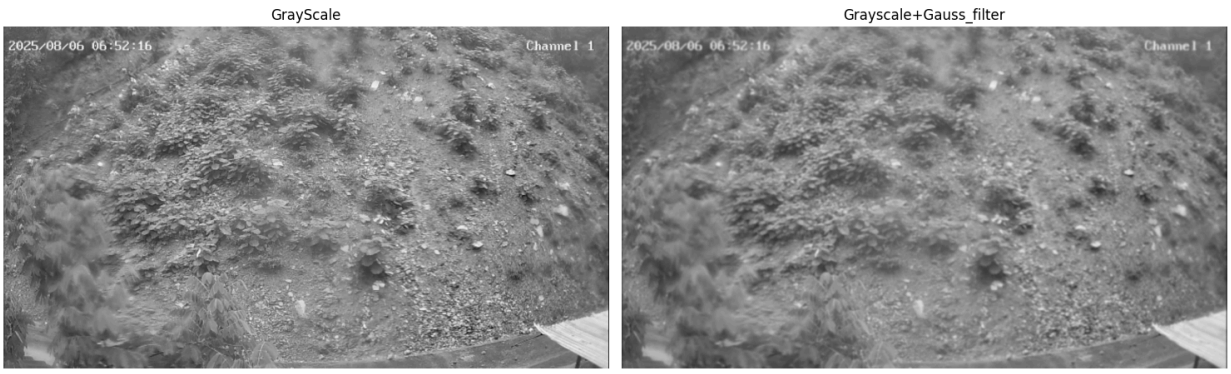
Gaussian Filtering (reduces the image noise):
The algorithm begins with Gaussian filtering to remove high-frequency noise while preserving edge information, this gives a slightly better accuracy while tracking displacements.
The two-dimensional Gaussian filter is mathematically defined as:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

where σ represents the standard deviation controlling the degree of smoothing. The filtered image I_f is obtained through convolution:

$$I_f(x,y) = I(x,y) * G_\sigma(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(u,v)G_\sigma(x-u,y-v)\,du\,dv$$

The discrete implementation employs separable filtering, reducing computational complexity from $O(n^2)$ to $O(n)$ per output pixel.
(Reduces complexity from O(N^2 K^2) to O(N^2 K) for N×N images with K×K kernels)



[For Demonstration, sigma(std.)=2.4 is used, though in practice sigma=0.8 is used]

Contrast Limited Adaptive Histogram Equalization (CLAHE): (Increases contrast while preventing noise amplification, enhances low contrast image for better accuracy)

CLAHE enhances local contrast by applying histogram equalization to individual image tiles while preventing noise amplification. The algorithm divides the image into $M \times N$ non-overlapping tiles and computes the cumulative distribution function (CDF) for each tile:[5][6][7]

$$\mathrm{CDF}(k) = \sum_{i=0}^{k} \frac{h(i)}{N_{\mathrm{pixels}}}$$

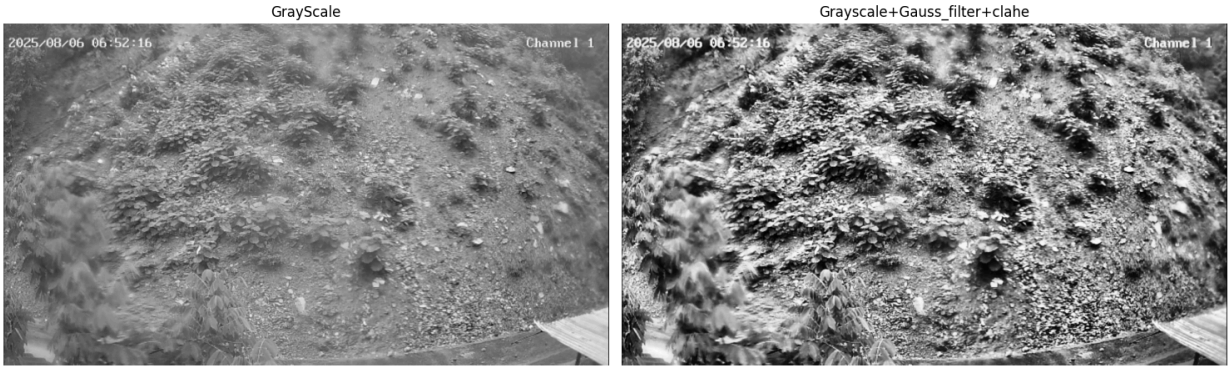
where $h(i)$ represents the histogram count at intensity level i. The contrast amplification is limited by clipping histogram values at a predefined threshold C_{limit} before CDF computation:[8][6][7]

$$h_{\mathrm{clipped}}(i) = \min(h(i), C_{\mathrm{limit}})$$

Excess histogram values are redistributed uniformly across all bins to preserve total pixel count. The transformation function becomes:[6]

$$T(i) = \mathrm{CDF}_{\mathrm{clipped}}(i) \times (\mathrm{L} - 1)$$

where L represents the number of intensity levels (256 for 8-bit images).

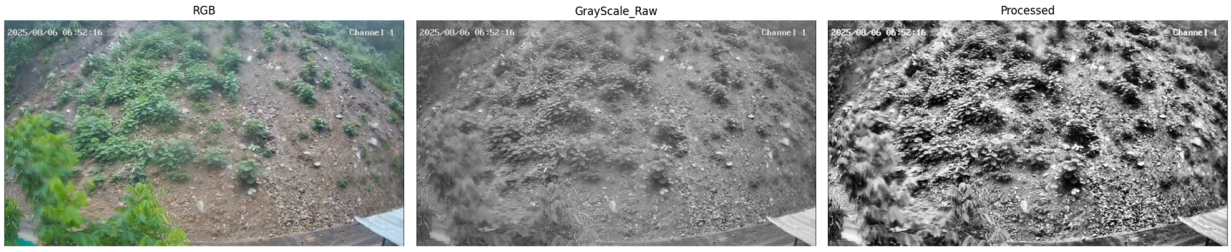


Histogram Stretching:

For low-contrast images (standard deviation < 30), histogram stretching enhances dynamic range by mapping pixel intensities to the full available spectrum. The mathematical transformation is:[10][11]

$$I_{\mathrm{stretched}}(x,y) = \frac{I(x,y) - I_{\mathrm{min}}}{I_{\mathrm{max}} - I_{\mathrm{min}}} \times (\mathrm{Max} - \mathrm{Min}) + \mathrm{Min}$$

where I_{min} and I_{max} are the 2nd and 98th percentile values respectively, and Max and Min represent the target intensity range (0 and 255 for 8-bit images). This percentile-based approach provides robustness against outliers

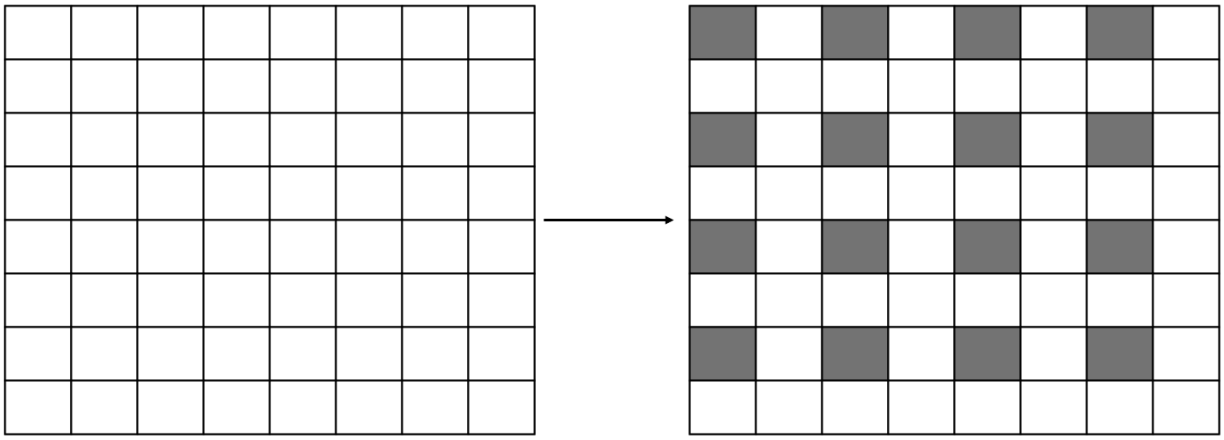


[Image after pre-processing]

Digital Image Correlation Algorithm

Partitioning the Images into smaller patches (of size subset_size*subset_size):
Subset Sampling,
The Image (both reference and deformed images) are partitioned into small Image Patches of size subset_size*subset_size, for step =1 every patch is selected, for step=2 every alternative patch is selected in both x and y axis respectively.

Using step size s, the algorithm analyzes $\lfloor (N-M)/s \rfloor^2$ subsets instead of $(N-M+1)^2$, reducing computational load by factor s^2.



Original Image

Partitioned Image, for step_size=2

where “grey squares” denote the selected Image Patches for further processing.

The Core DIC Algorithm:
Lets call the Image patches as subset or windows

The Centers of the Image patches/subsets/windows are stored in an array.
We Iteratively go through each subset in this array,
For a subset of center C in both reference and deformed images,
Normalize the subsets,
Win = (Win - Win.mean())/(Win.std() + 1e-8)

The core DIC algorithm employs FFT-based ***normalized cross-correlation (NCC)*** to measure similarity between reference and deformed image subsets. For two image windows f and g, the NCC is defined as:[13][14][8]

$$\mathrm{NCC}(u,v) = \frac{\sum_{x,y}\,(f(x,y)-\bar{f})(g(x+u,y+v)-\bar{g})}{\sqrt{\sum_{x,y}\,(f(x,y)-\bar{f})^2 \sum_{x,y}\,(g(x+u,y+v)-\bar{g})^2}}$$

where $\bar{f}$ and $\bar{g}$ represent mean intensities of the respective windows.[8][15]
The algorithm utilizes the convolution theorem for efficient computation:[16][15]

$$\mathrm{Corr}(f,g) = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}^{*}\{g\}\}$$

where $\mathcal{F}$ denotes the Fourier transform and $\mathcal{F}^{*}$ represents the complex conjugate. This approach reduces computational complexity from $O(N^4)$ to $O(N^2 \log N)$ for $N \times N$ image subsets.

This is done using scipy.signal library fftconvolve function with flipping the reference image using ref_win_norm[::-1, ::-1], a ***correlation matrix*** is created for this pair of subsets.

The ***correlation quality*** metric quantifies the reliability of the correlation peak:[2][18]

$$Q = \frac{\max(\mathrm{Corr})}{\sqrt{\sum\,(Corr_{\mathrm{norm}})^2 + \epsilon}}$$

where $Corr_{\mathrm{norm}}$ represent zero-mean normalized window intensities. This metric ranges from 0 to 1, with higher values indicating more reliable correlations.

The correlation quality is used to filter the regions to get reliably accurate displacements. Followed by a subpixel displacement refinement to get accuracy finer than the pixels, before finally computing the displacement vectors as mentioned in the Filtering and Refinement Sections.

For the ***computation of displacement vectors***, the displacement vectors (U,V) are computed as the difference between the refined peak location and the correlation matrix center:

$$U_{i,j}$$
$$V_{i,j}$$

where the center coordinates correspond to zero displacement. These vectors represent pixel-level displacements between reference and deformed states.\

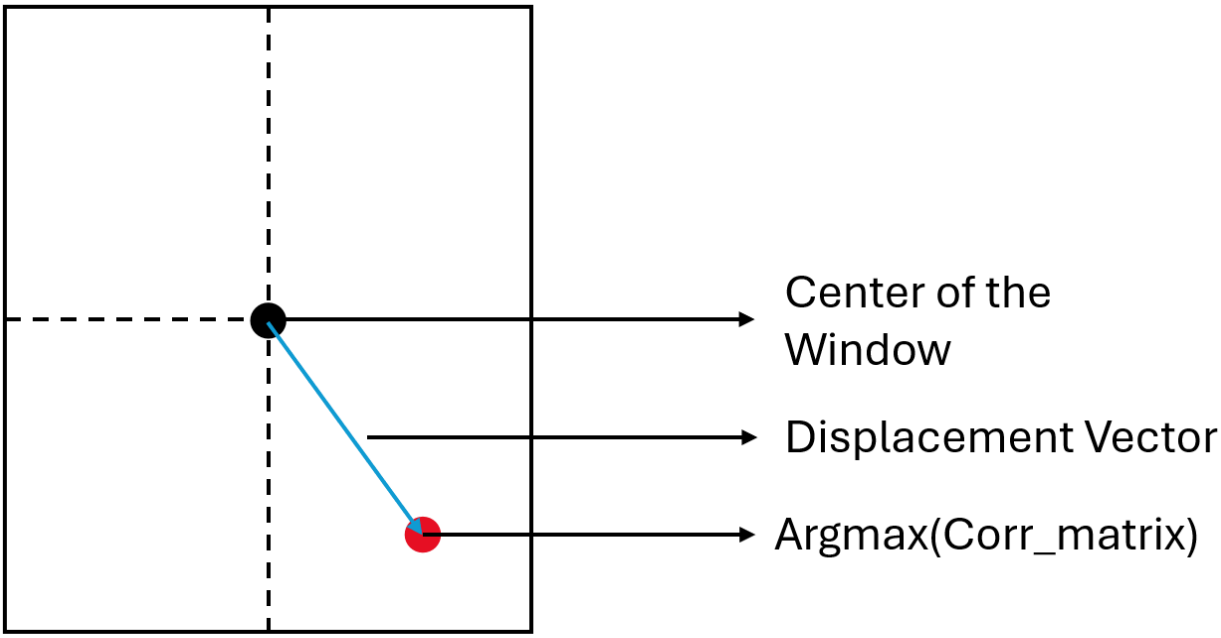
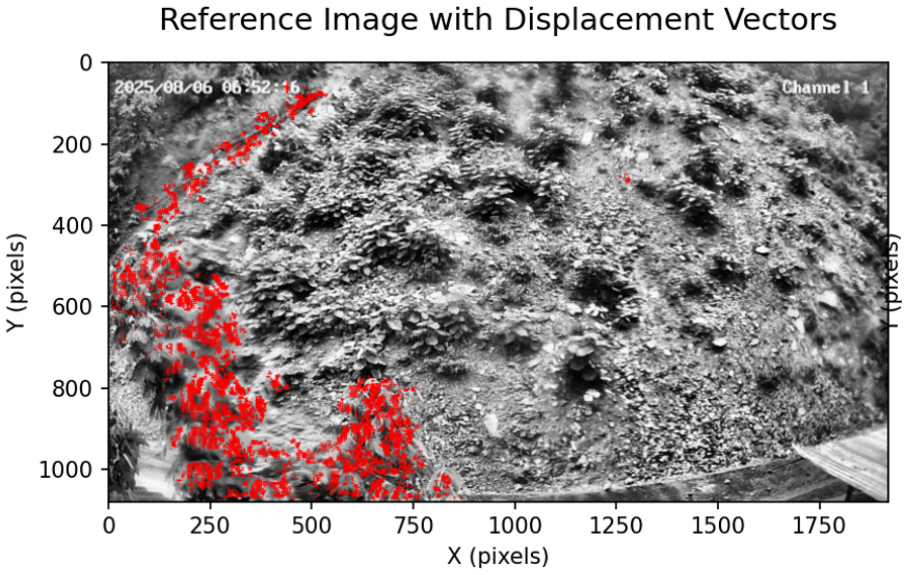
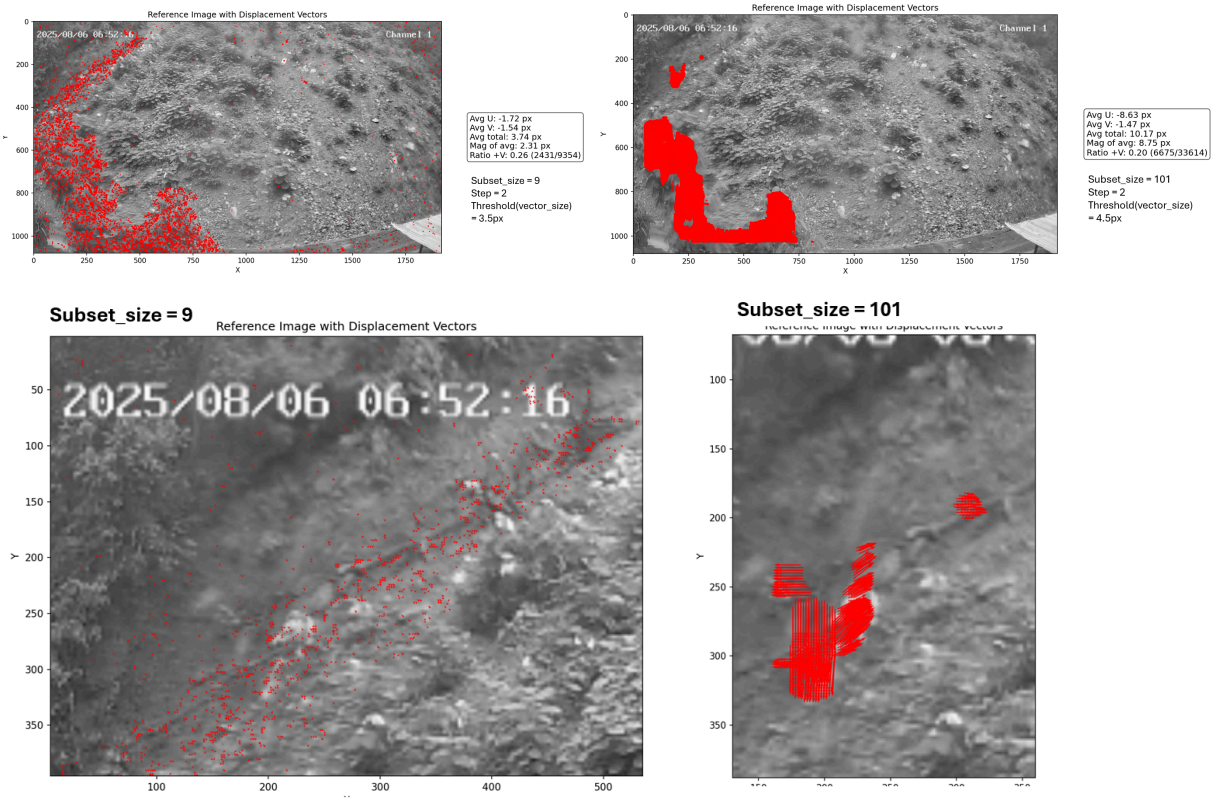


Image Subset

```
Avg U: -4.07 ± 6.93 px
Avg V: -1.17 ± 4.88 px
Avg total: 9.23 px
Mag of avg: 4.23 px
Ratio +V: 0.37 (1470/3948)
```



[Result]



[The above Images demonstrate the effect of changing subset_size,
Top-left Image: subset_size=9;
Top-right Image: subset_size=101;
Bottom Image: Zoomed-in view of Both the Images]

Increasing the Subset_size can give more accurate results, as it can track larger displacements but is more prone to error.

Refinement

Subpixel Displacement Refinement:
Gaussian Peak Fitting
To achieve subpixel accuracy, the algorithm fits parabolic curves to correlation peaks in both spatial directions. For the x-direction, given correlation values at positions $x-1$, x, and $x+1$:

$$C(x) = ax^2 + bx + c$$

The coefficients are determined through least squares fitting:

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix} = \begin{bmatrix} C(x-1) \\ C(x) \\ C(x+1) \end{bmatrix}$$

The subpixel peak location is obtained from the vertex formula:[20][19]

$$x_{\text{peak}} = -\frac{b}{2a}$$

provided that $a < 0$, ensuring a local maximum[19]. The condition on the Intensity $|x_{\text{peak}}| \leq 1.0$ validates the refinement within the neighborhood.

(More complex functions can be fit, and fitting on points greater than 3, to achieve finer accuracy, this can be used for high-res images where sub-pixel precision may matter. Although, in our experiments, it is used only as a demonstration.)

Filtering

Filtering very low-contrast subsets:
The algorithm implements texture quality assessment to exclude low-information regions. A texture threshold of 5.0 is applied to the standard deviation of normalized window intensities:

$$\sigma_{\text{texture}} = \sqrt{\frac{1}{N}\sum_{i,j}\left(\frac{I(i,j)-\mu}{\sigma+\epsilon}\right)^2}$$

where $\epsilon = 10^{-8}$ prevents division by zero. Windows failing this criterion are excluded from correlation analysis

Correlation-Quality Filter:

$$\text{Mask}_1 = Q_{i,j} > \theta_{\text{corr}}$$

where $\theta_{\text{corr}} = 0.3$ represents the correlation quality threshold.

1. Intuition for this/ What are few possible interpretations of this:
   They're probably not the same region.(reasons: images of not the same site or Or, the Camera has moved too much/ Images aren't aligned/ registered)
2. The subset in the reference and deformed images are so different that the displacement calculation will be very inaccurate. (Try with images that are temporaly less apart)

Adaptive Magnitude Filter:
The displacement magnitude is computed as:

$$M_{i,j} = \sqrt{U_{i,j}^2 + V_{i,j}^2}$$

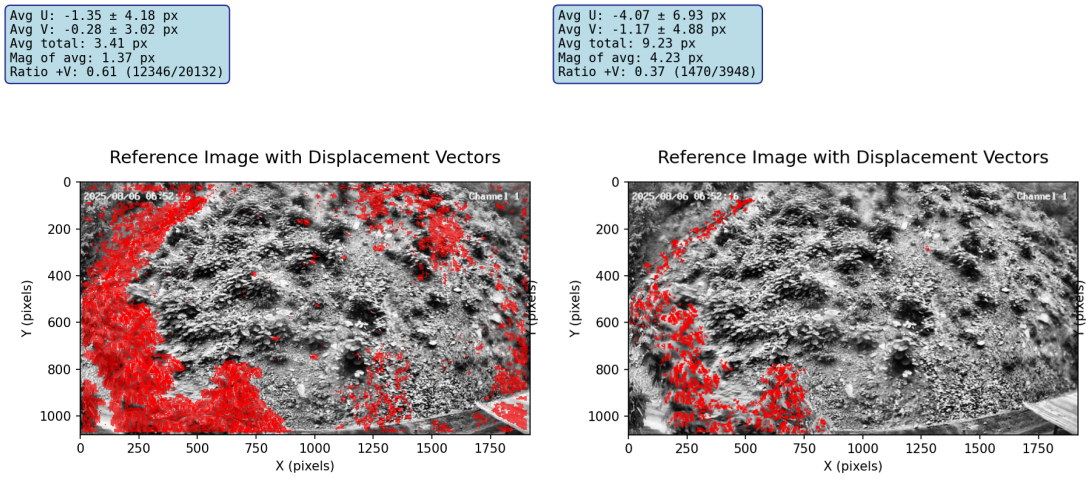
An adaptive threshold based on the 95th percentile of valid magnitudes is applied:

$$\theta_{\text{mag}} = \text{P}_{95}(M_{valid})$$

where M_{valid} includes only magnitudes passing the correlation quality filter.
Final Filter,

$$\text{Mask}_{\text{final}} = \text{Mask}_1 \wedge (M_{i,j} > \theta_{\text{mag}})$$

This ensures selection of high-quality, significant displacement vectors.



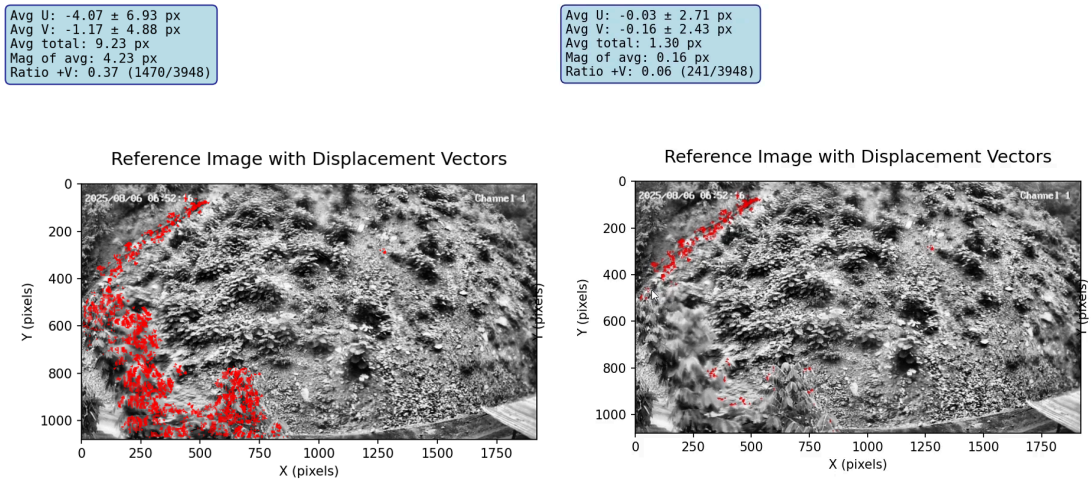
[Left Image: correlation_threshold=0.1, valid magnitude>75th percentile;
Right Image: correlation_threshold=0.3, valid magnitude>95th percentile]

Vegetation Segmentation Integration

The algorithm incorporates vegetation masking to exclude regions with significant vegetation movement from displacement analysis. For pixels where the vegetation mask indicates vegetation presence (pixel value = 255):

$$U_{\text{masked}}(i,j)$$
$$V_{\text{masked}}(i,j)$$

This approach ensures that displacement measurements focus on rigid terrain features rather than flexible vegetation

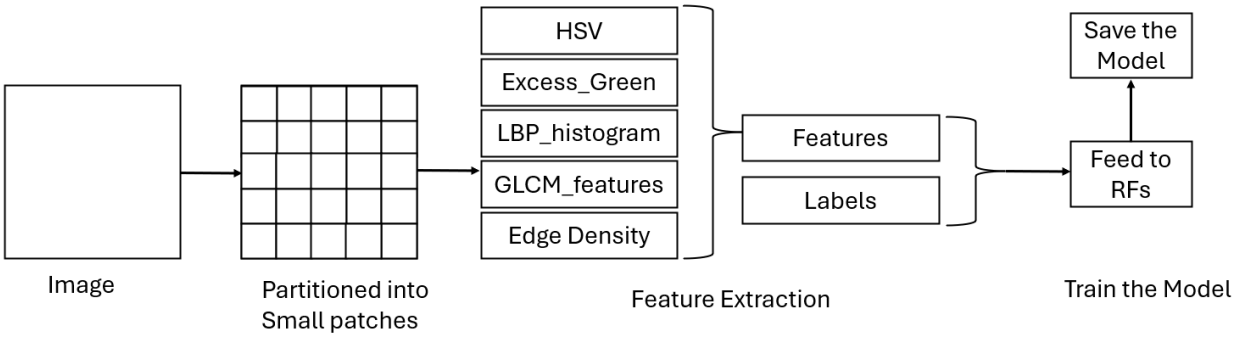


[Left Image: Displacement Vectors are Computed;
Right Image: Vegetation Mask is applied on the computed Displacement Vectors]

Vegetation Segmentation

Overview:

The system follows a patch-based approach where large images are divided into manageable square regions, each analyzed through multi-modal feature extraction before classification via Random Forest ensemble learning.



Features

The feature extraction component, computes a comprehensive 45-dimensional feature vector from each image pair.

RGB Statistics: For each color channel $c \in \{R, G, B\}$, the system computes:

·Mean: $\mu_c = \frac{1}{s^2} \sum_{x,y} P_c(x, y)$

·Standard deviation: $\sigma_c = \sqrt{\frac{1}{s^2} \sum_{x,y} (P_c(x, y) - \mu_c)^2}$

RGB feature vector: $\mathbf{f}_{\text{RGB}} = [\mu R, \sigma R, \mu G, \sigma G, \mu B, \sigma B]^T \in \mathrm{R}_6$

HSV Transformation: The RGB-to-HSV conversion follows the mathematical formulation:

$$H = \begin{cases} 0° & \text{if } \Delta = 0 \\ 60° \cdot \left(\frac{g - b}{\Delta} \bmod 6 \right) & \text{if } C_{\max} = r \\ 60° \cdot \left(\frac{b - r}{\Delta} + 2 \right) & \text{if } C_{\max} = g \\ 60° \cdot \left(\frac{r - g}{\Delta} + 4 \right) & \text{if } C_{\max} = b \end{cases}$$

where $C_{\max} = \max(r, g, b)$, $C_{\min} = \min(r, g, b)$, and $\Delta = C_{\max} - C_{\min}$.

HSV feature vector: $\mathbf{f}_{\text{HSV}} = [\mu H, \sigma H, \mu S, \sigma S, \mu V, \sigma V]^T \in \mathrm{R}6$

Excess Green Index: This vegetation index emphasizes green vegetation pixels using the formula:

$$\mathrm{ExG} = 2g - r - b$$

The Excess Green index is particularly effective for discriminating vegetation from soil and background materials in agricultural and environmental monitoring applications.

Local Binary Patterns (LBP): Uses uniform LBP with parameters $P = 24$ neighbors and radius $R = 3$:

$$\mathrm{LBP}_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(I_p - I_c) \cdot 2^p$$

where $s(x) = 1$ if $x \geq 0$, otherwise $s(x) = 0$. Uniform patterns contain at most two bitwise transitions and provide rotation-invariant texture description.

Gray-Level Co-occurrence Matrix (GLCM): The system computes six Haralick texture features from the co-occurrence matrix:

$$\mathrm{GLCM}_{d,\theta}(i,j) = \#\{(x,y): I(x,y) = i, I(x + d\cos\,\theta, y + d\sin\,\theta) = j\}$$

Features include contrast, correlation, energy, homogeneity, dissimilarity, and Angular Second Moment (ASM)

Motivation for Using Random Forests

Superior Performance on Small to Medium Datasets:
Leveraging Random Forest's ability for performing good on classification tasks with input as structured, tabular feature representations without requiring massive amounts of training data like the neural networks.
In our Implementation, the custom dataset we created had only ~14K datapoints (labeled image patches) but the model was still able to provide satisfactory results.

Also, it has the added benefit of Robustness to Noise, Outliers, and Missing Data and No Extensive Hyperparameter Tuning Required.

Computational Efficiency:
Random Forests offer an efficient balance between accuracy and computational cost. Both training and inference can be performed effectively on standard CPUs, achieving competitive results compared to deep neural network–based approaches while eliminating the need for GPUs. This makes them particularly suitable for deployment in resource-constrained environments.

Although several vegetation segmentation techniques utilize deep neural networks, such models typically require substantial computational resources and are less suited for real-time applications. RF model was able to classify the image in ~1-3 seconds.

In contrast, our approach intentionally adopts a simpler and smaller model, prioritizing computational efficiency and real-time feasibility over achieving absolute state-of-the-art accuracy. This design choice represents a deliberate trade-off between performance and practicality.

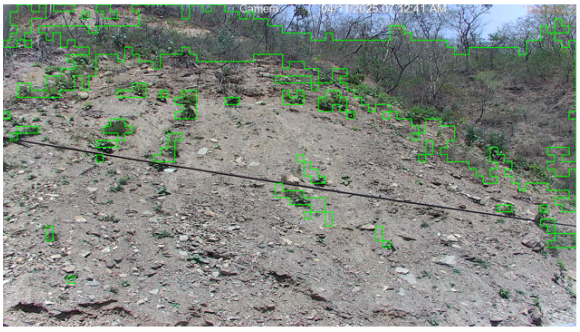
Comparative Analysis:
We experimented with multiple vegetation segmentation techniques before selecting the final model. Simpler methods such as **Graph-Cut Segmentation** and **K-means Clustering** failed to effectively differentiate vegetation in the given images, primarily due to their limited ability to handle color and texture variations in natural scenes.

In contrast, more complex models based on **Convolutional Neural Networks (CNNs)**—such as **Modified U-Net**, **HRNet**, and **YOLOv11 Segmentation**—are known to produce highly accurate vegetation segmentation results when properly trained and optimized.
However, these approaches typically incur significantly higher computational costs, with inference times often ranging from **several minutes per image**. Additionally, such deep models require large, well-annotated datasets for effective training, making data preparation a major practical challenge.

Among simpler methods, **classification based on HSV color space** performed best, offering very low computation time. However, HSV-based segmentation alone was inadequate for robust vegetation differentiation.
This **limitation** arises because certain vegetation regions appear brownish or greyish due to lighting, weather conditions, or camera calibration errors. Additionally, occluded or sparsely vegetated areas often contain mixed color ranges, complicating pure color-based segmentation.

The Random Forest model demonstrated superior capability in classifying vegetation under varying conditions. It effectively handled color and texture variations, accurately distinguishing vegetation from non-vegetative regions. Consequently, while HSV-based segmentation produced inconsistent results, Random Forest segmentation offered more stable and reliable performance with minimal computational costs

HSV-based Segmentation



Random Forests Segmentation

.

Training Random Forests

Patch Partition: The input image was partitioned into smaller patches to facilitate localized feature extraction. In our implementation, the reference image was divided into approximately 3,500 patches, and the aforementioned features were extracted from each patch to construct the dataset.

A custom **Image Annotator Tool** was developed to assist in dataset creation. This tool, available in the project's GitHub repository under /Image_segmentation/Dataset/Img_annotator_tool , allows manual annotation of vegetation regions. It automatically partitions the image into smaller patches (approximately 3,500 in our setup, or a patch size of 30 × 35 pixels, whichever is lower).

Using this approach, we generated a dataset containing roughly **14,000 data points**. The Random Forest (RF) model demonstrated satisfactory performance with this dataset. However, performance is expected to improve further with a larger and more diverse dataset (on the order of ~100,000 data points).

Training and Prediction

The extracted dataset was used to train a Random Forest classifier comprising an ensemble of 200 decision trees. The model was trained using the RandomForestClassifier implementation from the *scikit-learn* library (sklearn.ensemble).

Model evaluation and prediction were carried out using standard *scikit-learn* testing procedures. The trained model effectively segmented vegetation regions within the test images, validating the efficacy of the patch-based Random Forest approach.

Image | **Generated Segmentation Mask** | **Overlay Visualization**

[Vegetation Segmentation Pipeline]

[Results]

Post-Processing

Morphological Operations

The post-processing pipeline employs mathematical morphology to refine the initial classification results. The small object removal operation uses connected component analysis with an adaptive threshold:

$$\text{min\_area} = \frac{W \times H}{3500} \times 2$$

Morphological opening and closing operations follow the mathematical definitions:

- Opening: $A \circ B = (A \ominus B) \oplus B$
- Closing: $A \cdot B = (A \oplus B) \ominus B$

where $\ominus$ denotes erosion and $\oplus$ denotes dilation with structuring element B.

HSV-Based Region Refinement

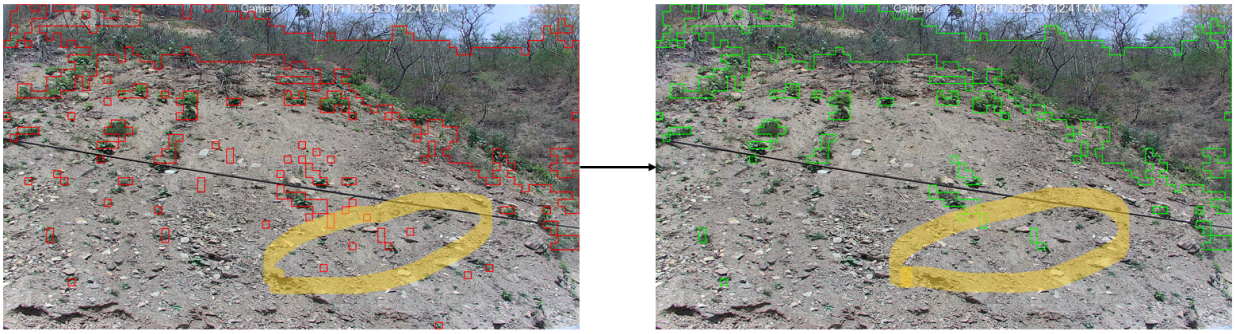
The final refinement stage applies vegetation-specific color constraints based on HSV statistics.

For each connected component R_k, the system computes mean hue and saturation values, applying the decision rule:

$$\text{Keep } R_k \text{ if } (|R_k| \geq A_{\min}) \wedge (35° \leq \mu_h \leq 85°) \wedge (\mu_s > 50)$$

For hue calculations, circular statistics are employed since hue is a circular variable:

$$\mu_h = \arctan\ 2(\sum \sin(h_i), \sum \cos(h_i))$$



Without Post-Processing **With Post-Processing**

Result

The proposed integration of **Digital Image Correlation (DIC)** with **Random Forest–based vegetation segmentation** was evaluated on sequential terrain images captured by a fixed-position camera system.

The results show that using digital Image Correlation was correctly able to track the displacement of regions with change and the vegetation masking significantly improves DIC reliability by filtering vegetation-induced motion artifacts.

Limitations

For DIC, some DIC displacement vectors were oriented opposite to the actual motion.\
For vegetation segmentation, model performance was limited by the size and diversity of the training dataset, leading to occasional misclassification of vegetation under atypical lighting or seasonal changes.

Conclusion

This work presents a computationally efficient landslide detection framework that combines DIC-based motion estimation with Random Forest–based vegetation filtering. The system effectively suppresses vegetation-induced noise, enhancing the accuracy and stability of displacement detection.
Its low computational demand enables real-time deployment on CPU-based systems, offering strong potential for low-cost, automated landslide monitoring.

Extras:

[not to be included in the final draft]

Image Noise:

Image noise refers to random variations of brightness or color in an image — basically, unwanted information that makes the picture look grainy, speckled, or distorted.
Causes of Image Noise: Sensor limitations, Thermal noise (from electronic circuits), Various limitations of digital camera, compression of digital images.
High-frequency noise means the noise fluctuates **rapidly** across neighboring pixels — creating sharp, grainy patterns,
Low-frequency components → smooth variations (like sky or skin tones)
High-frequency components → rapid changes in intensity (like edges, textures, or fine details)

Dynamic Range:

Dynamic range refers to the ratio between the brightest and darkest parts of an image that can be captured or displayed.

Dynamic range is usually expressed as a **ratio** or in **stops (EV)** (each stop = doubling of light intensity):
Dynamic Range= Maximum measurable light intensity/Minimum measurable light intensity
For example:
Camera sensor: 1:20,000 (≈14 stops)
Human eye: ~1:1,000,000 (≈20 stops)
Typical monitor: 1:1,000 (≈10 stops)

Contrast:

Contrast is a measure of intensity variation in an image.
Standard deviation of pixel intensities is one of the simplest quantitative measures of contrast.
Low std⇒Low variation in intensities⇒Low contrast region
High std⇒High variation in intensities⇒High contrast region