EGO-VC – Evolutionary GPU-Optimization of Visual Correspondences for Image Alignment

Thomas Vincent Chang*[1]
thomas.chang@
th-nuernberg.de

Kay Hartmann*[1]
hartmannka80488@
th-nuernberg.de

Bastian Kuth[2]
Bastian.Kuth@hs-coburg.de

Simon Seibt[3]
simon.seibt@hs-kempten.de

Bartosz von Rymon Lipinski[1]
bartosz.vonrymonlipinski@
th-nuernberg.de

[1]Nuremberg Institute of Technology, Kesslerplatz 12, 90489 Nuremberg, Germany
[2]Coburg University, Sonntagsanger 1, 96450 Coburg, Germany
[3]Kempten University of Applied Sciences, Bahnhofstrasse 61, 87435 Kempten, Germany

ABSTRACT

Many computer vision applications, such as panoramic stitching, image morphing, and image registration depend on precise feature correspondences. While current machine-learning-based methods excel at detecting precise feature matches, they do not necessarily guarantee visual alignment. This work presents an evolutionary, mesh-based optimization framework that enhances alignment quality and improves visual coherence as a postprocessing step for any feature-matching algorithm: The approach employs the well-known *enhanced correlation coefficient* (ECC) as a visual error metric, efficiently computed in parallel using the rasterization capabilities of modern graphics hardware. Given any state-of-the-art feature matcher, the proposed method constructs a Delaunay feature point mesh and evolutionarily refines the ECC alignment of each generated triangle. This results in a more precise registration beyond the accuracy of initial feature matches. Extensive evaluation across multiple standard image datasets confirms the proposed method's effectiveness, yielding ECC improvements up to 18.3% and ensuring high visual quality in downstream applications, particularly in challenging image areas with occlusions. For morphing-based applications, this leads to sharper, smoother transitions between image pairs, minimizing visual artifacts.

Keywords

Computer Vision, Image Alignment, ECC, Image Correspondences, Feature Matching, Evolutionary Optimization

1 INTRODUCTION

Precise feature correspondences and accurate image alignment are fundamental to numerous computer vision applications, including panoramic stitching [Wan24, Hua24], image morphing [Sei23], and image registration [Dar24, Wan24]. While dense feature point matches can be used to construct a mesh of corresponding triangles, effectively representing local image region alignments, current feature matching techniques often prioritize matching accuracy over the coherence of image content between matched points. This can lead to inconsistencies and suboptimal visual alignment. To address this, we propose an evolutionary optimization method that refines the positions of matched feature points within a mesh for visual triangle alignment. This approach ensures higher visual quality for downstream tasks, especially in challenging areas with occlusions, resulting in sharper and smoother morphing transitions and more robust registration.

This work makes the following contributions:

- Development of an evolutionary algorithm to optimize mesh-based alignment using the *enhanced correlation coefficient* (ECC) image alignment metric.
- Parallelization of the ECC error norm calculation through decomposition into multiple summations.
- Efficient GPU-based ECC evaluation of positional feature point mesh mutations.

We validate our technique through quantitative measurements and qualitative evaluations, demonstrating its potential to improve visual quality for image stitching, morphing, and registration. A runtime analysis further highlights the efficiency of the proposed approach as a post-processing step applicable to any state-of-the-art feature matcher for image pairs.

2 RELATED WORK

Image alignment and feature matching are cornerstone tasks in computer vision, with ongoing research continually pushing the boundaries of performance and robustness. This section reviews recent advances in these areas, focusing on deep learning-based feature matching techniques, image alignment strategies, and relevant evaluation metrics.

Feature Matching

Recent advances in feature matching have shifted from traditional methods like SIFT [Low04], SURF [Bay06], and ORB [Rub11] to deep learning-based approaches, particularly transformer architectures. Models such as ASpanFormer [Che22], ECO-TR [Tan22], LightGlue [Lin23], and LoFTR [Sun21] have significantly improved matching accuracy, especially in challenging scenarios. However, these methods typically prioritize point-wise accuracy, often overlooking the visual coherence of the resulting alignments of image regions between matched feature points.

Our work addresses this by introducing a post-processing step that refines feature matches based on visual coherence of this intermediate image content.

Image Alignment Techniques

Deep learning has significantly advanced image alignment, with methods like Deep Image Homography Estimation [Ngu18], PDC-Net [Hua22], deep implicit keypoints [Sar20], and recurrent geometric transformations [Zha21] leveraging convolutional and transformer networks to achieve improved alignment.

These techniques typically learn global or local parametric transformations, while our approach directly optimizes individual feature match positions within a predefined mesh for more flexible adjustments and potentially improved visual coherence.

Image Alignment Metrics

A variety of metrics exist for evaluating image similarity, each with trade-offs between sensitivity to image variations and computational efficiency. Traditional metrics like Sum of Squared Differences (SSD) and Normalized Cross-Correlation (NCC) offer simple pixel-wise comparisons but can be sensitive to illumination and viewpoint changes [Bro92]. The Enhanced Correlation Coefficient (ECC) provides greater robustness to such variations, making it well-suited for image registration tasks [Eva08]. Structural Similarity Index (SSIM) [Wan04] incorporates structural information, potentially capturing subtle misalignments that affect visual coherence. While Peak Signal-to-Noise Ratio (PSNR) is commonly used for image quality assessment, it may be less sensitive to misalignments.

Deep learning-based metrics, such as perceptual loss [Joh16] and LPIPS [Zha18], offer potential advantages in capturing perceptual similarity and handling complex transformations. However, these methods generally come with a higher computational cost.

Our approach prioritizes ECC as the optimization objective due to its established robustness for image alignment and suitability for parallel computation on GPUs. This choice allows to effectively refine feature matches while maintaining computational efficiency.

3 METHOD

To optimize the visual alignment of two images, a set of dense feature point correspondences is first computed using any state-of-the-art feature matcher. These correspondences are then used to construct a robust and consistent mesh of corresponding Delaunay triangles. The visual alignment of each triangle pair is evaluated using the ECC error norm. This section details the proposed algorithm, beginning with the evolutionary optimization process employed to refine the mesh alignment. The GPU-accelerated implementation of the ECC error norm calculation is then elaborated, which forms the basis for the efficiency of the approach.

Optimization Algorithm

This section describes the optimization algorithm employed to refine the alignment of a mesh constructed from feature point correspondences across n images.

First, a consistent set of feature points P is computed across all n images, ensuring that the number of points is consistent ($|P_0| = |P_1| = \cdots = |P_{n-1}|$), with P_i representing the positions of each point $p \in P$ in image I_i. For the case of two images, this can be achieved using any state-of-the-art feature matcher, such as those discussed in Section 2.

Following Delaunay triangulation, P_0 is triangulated to form a mesh T, which serves as a common basis for all other sets of feature points P_i with $i > 0$. This ensures that each triangle $t \in T$ shares the same vertex feature points $p \in P$ across all images. For instance, if triangle $t^0 \in T$ is defined by vertices $\{p^a, p^b, p^c\}$, where p^a, p^b, and $p^c \in P$, then in each image I_i, the corresponding triangle $t_i^0 \in T_i$ will have vertices $\{p_i^a, p_i^b, p_i^c\}$, with p_i^a, p_i^b, and $p_i^c \in P_i$.

The optimization of the entire mesh proceeds according to the procedure outlined in Algorithm 1. For each feature point $p_i^j \in P_i$, a random positional adjustment within a radius r is calculated. This adjustment is referred to as a mutation M_i^j for p_i^j. Each mutation M_i^j affects the triangles $T_{p_i^j} \subseteq T_i$ for which p_i^j is a vertex. Note that this mutation does not influence the feature point positions in any other image.

If the average *ECC* of all triangles $t_{p_i^j} \in T_{p_i^j}$ after applying mutation M_i^j is higher than the average *ECC* before the mutation, then M_i^j is considered to improve the image alignment and is applied to p_i^j. In the case of two images, the *ECC* is calculated for each pair of corresponding triangles $\{t_{p_0^j}, t_{p_1^j}\}$. If $n > 2$, the *ECC* is calculated pairwise between I_i and all other images.

To optimize the total mesh T, these mutations are applied to each image consecutively. For each $p_i^j \in P_i$, a set of m mutations is calculated. If the best mutation M_{best} in terms of *ECC* improvement exceeds the *ECC* of the original feature point position p_i^j, then it is applied and overrides p_i^j. This process is performed iteratively, with each iteration involving the mutation of each image. After each iteration, the radius r decays by a factor of $d \in [0;1]$ to ensure convergence. The procedure continues until the total *ECC* improvement across all images falls below a predefined threshold t.

To prevent mesh degeneration, the radius r is limited to ensure that a point p cannot move outside its enclosing triangles. Therefore, the minimum distance between p and its opposing edges is determined before each mutation, and r is adjusted accordingly, as illustrated in Figure 1 (a). To reduce runtime overhead, triangles connected to each feature point are precomputed.

```
Algorithm 1: Optimize Mesh Alignment
Input : Matched feature point positions
        $P = \{P_0, P_1, \dots, P_{n-1}\}$ for $n$ images,
        Mesh triangles $T$,
        Iteration termination criteria $t$,
        Maximum mutation radius $r$,
        Mutation radius decay $d$,
        Mutation instance count $m$,
        $ECC$ for a set of triangles
Output: Optimized feature point positions $P^*$
  $P^* \leftarrow P$ {Initialize best mutation set}
  $r' \leftarrow r$ {Initialize maximum mutation radius}
  $E' \leftarrow -1$ {Initialize previous cumulative ECC}
  $E_{imp} \leftarrow 1$ {Initialize previous ECC improvement}
  while $E_{imp} > t$ do
    $E \leftarrow 0$
    for all images $i = 0$ to $n-1$ do
      for all points $p \in P_i^*$ do
        $T_p \leftarrow \{t \in T \mid p \in t\}$
        Generate $m$ random mutations within $r'$:
        $M \leftarrow mutate(T_p, m, r')$
        $M_{best} \leftarrow \arg\max_{k \in \{0,1,\dots m-1\}}(ECC(M_k))$
        if $ECC(M_{best}) > ECC(T_p)$ then
          Apply mutation to point $p$:
          $p \leftarrow p + M_{best}$
          $E \leftarrow E + ECC(M_{best})$
        else
          $E \leftarrow E + ECC(T_p)$
        end if
      end for
    end for
    $r' \leftarrow r' \times d$ {Apply radius decay}
    $E_{imp} \leftarrow \frac{E}{E'} - 1$
    $E' \leftarrow E$
  end while
  return $P^*$ {Return optimized feature points}
```

GPU-Accelerated ECC as Error Norm

The optimization algorithm described in Section 3 involves numerous ECC calculations. This section details the GPU-accelerated and parallelized implementation of the ECC error norm computation, which significantly enhances the efficiency of the optimization process.

To facilitate parallelization, the ECC error norm is expressed as a series of sums:

$$E_{\mathrm{ECC}}(\vec{x}, \vec{y}) = \frac{n\widehat{xy} - \hat{x}\hat{y}}{\sqrt{(n\widehat{xx} - \hat{x}\hat{x}) \cdot (n\widehat{yy} - \hat{y}\hat{y})}}. \qquad (1)$$

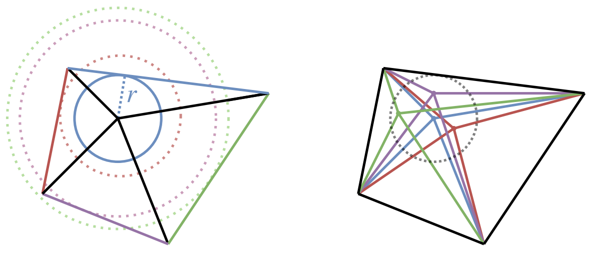
This representation allows the ECC computation to be encapsulated as a tuple $(\hat{x}, \hat{y}, \widehat{xx}, \widehat{yy}, \widehat{xy}, n)$, comprising six accumulators, collectively referred to as the *preECC*. This approach is inspired by the recommendations of Chan et al. [Cha83] for combining samples in variance calculations. By employing this analysis, the ECC computation is fully parallelized through the use of multiple independent *preECCs*, which are combined in a final, single-threaded step.

This parallelized ECC computation is then employed to evaluate whether a change in the position of a feature point p improves the alignment of the triangles T_p of which p is a vertex, as described in Section 3. Using *GPU instancing*, all triangles T_p can be rendered m times in a single draw call. The standard rendering pipeline, including both vertex and fragment stages, is utilized for error evaluation. Instead of rendering pixels to a framebuffer, the ECC is computed directly to assess alignment quality. The framebuffer size is set to match the input image dimensions, and multi-sampling is omitted, as testing indicated no practical benefits.

Implementation – Vertex Stage

- Source and target positions of feature points are passed as vertex attributes.
- The vertex undergoing mutation is offset to a pseudo-random position within a circle of radius r, centered around its current position. The random seed is derived from the instance identifier and a global seed provided as a uniform variable.
- The first instance position remains unchanged to include the original feature point in the evaluation.

- The radius r is constrained to prevent mesh degeneration, ensuring that p does not move outside the triangle spanned by T_p.



(a) Radius computation (b) Mutation instances

Figure 1: Feature point mutation: (a) Maximum mutation radius r defined by the closest opposing edge (blue). (b) Evaluating three mutations and the original position.

Implementation – Fragment Stage

In the fragment stage, both images are sampled using the feature point's positions (passed from the vertex shader as texture coordinates), and the *preECC* of the current fragment is computed by accumulating values from each color channel. A global buffer of size $m{\cdot}6$ stores the six *preECC* values per instance. These local *preECC* values are added to six global counters for the corresponding instances using synchronized atomic add operations to handle parallel fragment processing.

Unsigned integer arithmetic is used instead of floating-point arithmetic to avoid rounding errors and ensure deterministic *preECC* values, regardless of fragment addition order. For images with a color depth of 256, integer operations suffice without additional quantization.

To address potential overflow issues with 32-bit integers for large triangles, values are scaled by right-shifting them by $\max(0, \lceil \log_2(V) \rceil - 32)$ bits, where V is the worst-case maximum value:

$$V = (d-1)^2 \cdot c \cdot (n-1) \cdot A, \qquad (2)$$

where c is the number of color channels and A is the total triangle area. This scaling is reversed in floating-point arithmetic during the final ECC computation.

A compute shader with m threads processes the global *preECC* counters to compute the final ECC. This approach enables the evaluation of m mutations and the identification of the optimal feature point adjustment in just two draw calls: one for rendering and one for the compute shader. The evaluation of multiple color channels is achieved by concatenating the channels into a single vector. For alignment against multiple targets, the source vector $\vec{x}$ is repeated, and target values $\vec{y}$ are concatenated accordingly. The initial ECC value is expected to be positive, as the input mesh is pre-aligned.

To balance computational efficiency and hardware compatibility, the number of instances m can either remain constant based on the number of compute units of the hardware or dynamically scaled according to the area influenced by the current feature point:

$$m' = m \frac{r^{*2}}{r^2} \qquad (3)$$

where r is the decaying mutation radius per iteration, r^* is the capped mutation radius based on its enclosing triangles, and m is the global instance meta parameter.

All geometry data is stored on the GPU throughout the process, with updates applied only when a feature point's position is improved, ensuring efficient memory management and processing.

4 EVALUATION

This section presents a comprehensive evaluation of the proposed optimization technique, encompassing parameter exploration, benchmarking, and visual analysis. All experiments were conducted on a system equipped with an Intel i9-14900K CPU and an NVIDIA GeForce RTX 4070 SUPER GPU. As most feature matchers primarily support pairwise image correspondences, this evaluation focuses on optimizing feature matches between two images (i.e. $n = 2$).

Datasets

The evaluation is conducted across three distinct datasets to ensure a comprehensive assessment of the proposed optimization technique:

- **Oxford** [Mik05]: This dataset comprises image sequences with five different changes in imaging conditions: viewpoint changes, scale changes, image blur, JPEG compression, and illumination. Each test sequence contains six images with gradual geometric or photometric transformations at a medium resolution (approximately 800×640 pixels). Two scenes (zoom and rotation transformations) were excluded from the evaluation due to insufficient initial feature points provided by the feature matchers.
- **ETH3D** [Sch17]: A dataset designed for benchmarking and evaluating stereo and multi-view stereo algorithms. It includes high-resolution images of indoor and outdoor scenes, ranging from structured, man-made environments to natural, unstructured settings. The evaluation included 13 scenes consisting of between 14 and 76 images.
- **Custom EGO-VC Dataset**: This dataset consists of three scenes (two outdoor and one indoor) captured using a DJI Air 2S drone, incorporating varying degrees of camera movement, including both small and large changes in camera position and angle. Two of these scenes were used as training data for meta parameter optimization (see Section 4).

For the pairwise evaluation presented in Section 4, each sequence is divided into pairs of consecutive images.

Metrics

The effectiveness of the proposed optimization is quantified using the ECC metric, calculated as the average ECC value across all triangles within the mesh. This metric serves to evaluate alignment quality across different datasets and feature matchers, reflecting the improvement achieved through mesh optimization.

Meta Parameters

To optimize the performance of the mesh optimization algorithm (detailed in Section 3), a comprehensive meta parameter analysis was conducted using a training dataset comprising one indoor and one outdoor scene. This section presents the results of this analysis, examining the influence of the key meta parameters m, r, and d on both runtime and final mesh quality, as measured by ECC improvement. Figure 2 visualizes the performance of various parameter combinations, plotting average runtime against average ECC improvement. The optimization process was terminated when the relative ECC improvement per iteration fell below 0.5% (termination threshold $t = 0.005$). The parameter m was dynamically scaled based on Equation (3).

While the majority of configurations yielded similar final ECC values, increasing the number of mutation instances m per iteration resulted in slightly improved ECC, albeit with a proportional increase in runtime. Beyond $m = 256$, the gains in ECC were negligible compared to the added computational cost. The initial mutation radius r exhibited the most significant impact on alignment quality. Larger radii (up to $r = 10$) substantially improved convergence without affecting runtime. However, an unbounded radius ($r = inf$) led to premature and poor convergence. A decaying radius ($d < 1.0$) further enhanced convergence, particularly for smaller numbers of instances. A 50% decay rate ($d = 0.5$) consistently produced the best ECC improvements across all m values.

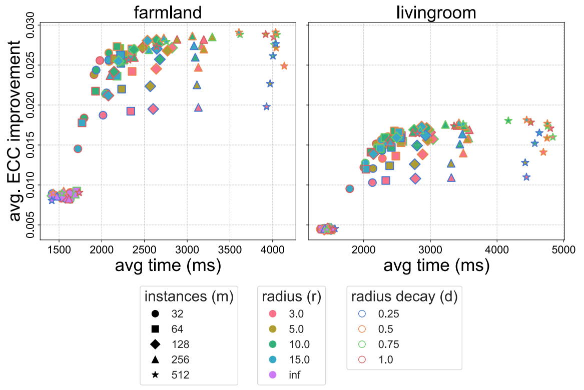


Figure 2: Meta parameter measurements.

Based on these observations, the following meta parameters were selected for all subsequent evaluations: $m = 256$, $r = 10$, and $d = 0.5$.

Quantitative Results

Table 1 presents the quantitative analysis of the achieved ECC improvement across different matching algorithms and datasets. For each combination of algorithm and dataset, the ECC image alignment is compared with and without the proposed optimization step. The total row provides the weighted average for each dataset based on scene sizes.

The proposed method consistently improves ECC values for all evaluated combinations of scenes and matching algorithms. The improvements range from 2.355% for ASpanFormer on the Oxford dataset to 12.912% for LoFTR on the ETH3D dataset. This highlights the significant impact of the post-processing step, particularly for cases with initially poor image alignment (i.e. $ECC < 0.8$), where the refinement contributes most substantially to enhancing visual coherence.

Qualitative Results

The visual impact of the proposed optimization technique on image alignment is illustrated in Figure 3. In these heatmaps, areas with low ECC alignment are represented with 'hotter' colors than regions with better alignment. Many unoptimized feature matching results exhibit weaknesses in properly aligning intricate, detailed structures, particularly those with repeated patterns and those encountering occlusions and parallaxes. By applying the proposed optimization post-processing, the majority of image regions demonstrate improved ECC alignment. For morphing-based applications, this translates to smoother transitions and blending in these challenging image regions.

Figure 4 further highlights the algorithm's ability to optimize visual alignment across diverse scene types and feature matchers. The optimization process effectively reduces or partially eliminates visual misalignments, enhancing the overall sharpness of the morphed images.

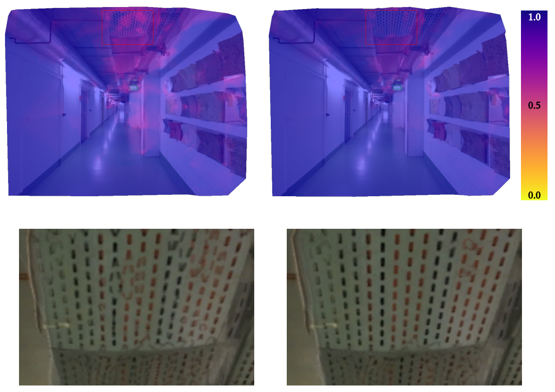
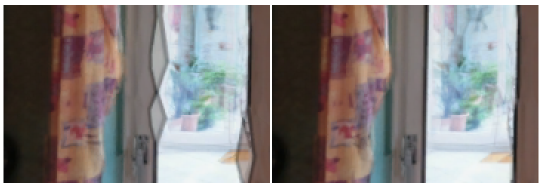
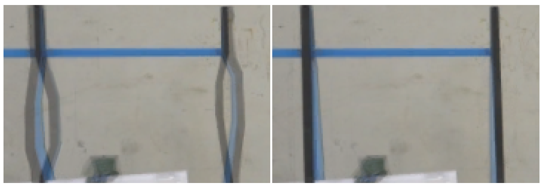


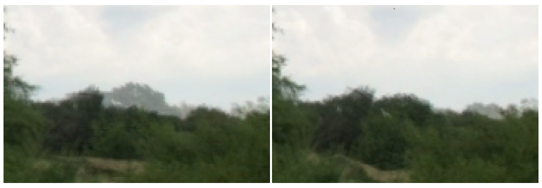
Figure 3: ECC heatmap with (right) and without (left) optimization, along with morphing-based RGB visualization.

Dataset		**ASpanFormer** [Che22]		**ECO-TR** [Tan22]		**LightGlue** [Lin23]		**LoFTR** [Sun21]	
		Mean ECC with (right) and without (left) optimization							
Oxford [Mik05]	Bikes	0.990	0.992	0.989	0.991	0.975	0.989	0.991	0.993
	Graf	0.955	0.970	0.848	0.986	0.940	0.987	0.826	0.906
	Leuven	0.960	0.971	0.968	0.979	0.946	0.977	0.957	0.972
	Trees	0.883	0.930	0.856	0.887	0.796	0.881	0.869	0.924
	UBC	0.837	0.890	0.868	0.889	0.798	0.859	0.861	0.900
	Wall	0.961	0.968	0.933	0.966	0.898	0.968	0.941	0.960
	Total	**0.934**	**0.956**	**0.910**	**0.950**	**0.892**	**0.944**	**0.909**	**0.944**
			+2.355%		+4.396%		+5.830%		+3.850%
ETH3D [Sch17]	Courtyard	0.761	0.842	0.760	0.847	0.816	0.904	0.682	0.801
	Delivery	0.827	0.870	0.884	0.934	0.820	0.903	0.818	0.879
	Electro	0.567	0.658	0.709	0.788	0.591	0.702	0.533	0.659
	Facade	0.757	0.848	0.757	0.839	0.821	0.894	0.685	0.812
	Kicker	0.743	0.821	0.673	0.753	0.769	0.870	0.640	0.757
	Meaow	0.925	0.954	0.748	0.776	0.886	0.926	0.822	0.873
	Office	0.838	0.891	0.515	0.563	0.758	0.834	0.770	0.834
	Pipes	0.694	0.783	0.806	0.885	0.638	0.743	0.672	0.777
	Playground	0.647	0.743	0.679	0.788	0.616	0.728	0.585	0.725
	Relief	0.888	0.921	0.911	0.952	0.852	0.915	0.883	0.930
	Relief_2	0.893	0.926	0.915	0.954	0.860	0.916	0.880	0.925
	Terrace	0.828	0.894	0.812	0.889	0.820	0.907	0.790	0.878
	Terrains	0.871	0.920	0.882	0.935	0.841	0.915	0.856	0.914
	Total	**0.776**	**0.844**	**0.780**	**0.848**	**0.777**	**0.860**	**0.728**	**0.822**
			+8.763%		+8.718%		+10.682%		+12.912%
EGO-VC	Farmland*	0.920	0.949	0.889	0.936	0.860	0.923	0.919	0.951
	Livingroom*	0.947	0.964	0.760	0.795	0.864	0.910	0.942	0.965
	Nature Walk	0.844	0.895	0.781	0.858	0.799	0.870	0.840	0.895
	Total	**0.907**	**0.938**	**0.787**	**0.839**	**0.840**	**0.898**	**0.902**	**0.938**
			+3.418%		+6.607%		+6.905%		+3.991%

Table 1: ECC improvement across datasets and methods. *Farmland and livingroom were used for meta parameter training.


(a) Grand Living Room - LightGlue


(b) Delivery - LightGlue


(c) Farmland - ECO-TR

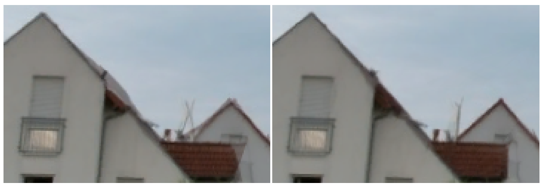

(d) Farmland - ECO-TR

Figure 4: Visual comparison with (right) and without (left) optimization. Selected areas from different datasets before and after applying the proposed optimization technique. Each image shows the intermediate morph when aligning two images.

Runtime

Figure 5 illustrates the relationship between the number of matched feature points and execution time across different image resolutions. Data points are colored by resolution, with darker shades representing higher resolutions. Note that some feature matchers internally scale input images, resulting in multiple resolutions per dataset. To exclude exceptionally fast executions due to poor matches and early termination, we filtered out image pairs with fewer than 200 matches or an initial ECC below 0.7. A moving average with a window size of 10 was applied to smooth the scatter plot.

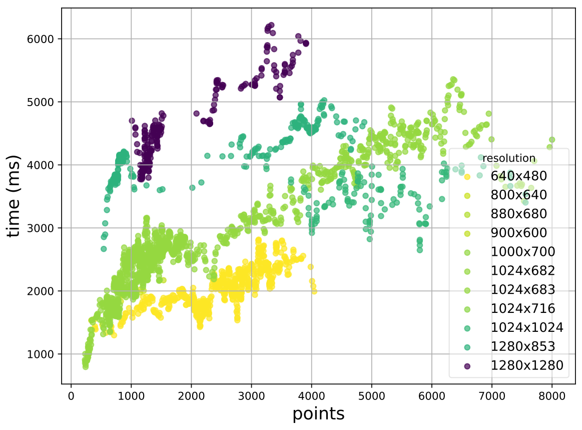


Figure 5: Runtime across resolutions and point densities.

As expected, execution time generally increases with both image resolution and the number of initially matched feature points. For the lowest resolution (640×480 pixels), optimization time ranges from 1.8 seconds for 250 points to 2.3 seconds for a maximum of 3000 points, with 2.1 seconds for an average of 2200 points. At 1024×1024 resolution, optimization takes $3.2 - 3.8$ seconds (for $4000 - 6000$ matches) and averages 3.5 seconds for approximately 5100 points.

The post-processing time represents a significant overhead compared to the initial feature matching time. Across the evaluated scenes, this overhead is at least 112%. For instance, with image pairs at 1280×853 resolution, ECO-TR detects an average of 3649 matches in approximately 4.0 seconds, which are then refined in 4.5 seconds, achieving a 9.87% ECC improvement. For LoFTR, which detects an average of 944 matches in 0.16 seconds at 640×480 resolution, the optimization takes 2.2 seconds, resulting in an 18.3% ECC improvement at a runtime overhead of 1375%. For latency-insensitive applications that require high alignment accuracy, such as medical image registration, this trade-off between ECC improvement and the additional computational cost may be justified.

Limitations

Despite the improvements achieved, our method still faces limitations in regions with deficient input geometry. These challenging scenarios include areas with significant parallaxes, sparse feature matches, and image elements visible in only one image of the pair. As illustrated in the bottom section of Figure 3, some visual misalignments could not be unresolved. Furthermore, optimizing one region can sometimes negatively impact the alignment of adjacent areas, as shown in Figure 4c.

5 CONCLUSION

This work presented a novel method for optimizing the visual alignment of image pairs. Our approach leverages an evolutionary algorithm for positionally adjusting a mesh of feature matches, using the ECC as the evaluation criterion, and is specifically designed for efficient parallel computation on GPUs. The evaluation demonstrated a significant improvement in ECC, with average improvements reaching up to $9.87\% - 18.3\%$ across the tested datasets. Visual comparisons of morphed images further confirm the effectiveness of our method, revealing a reduction in artifacts and increased sharpness after optimization. While our method achieves substantial improvements, optimization may be less effective for regions with deficient input geometry, such as significant parallaxes, sparse feature matches, or image elements visible in only one view. Future work includes an extendes evaluation framework for multi-image alignment (i.e., $n > 2$). Additionally, investigating alternative optimization strategies, such as gradient descent, and exploring alternative image similarity metrics, such as SSIM, PSNR, LPIPS and other machine learning-based loss functions, offers potential for further enhancing alignment quality and computational efficiency.

Acknowledgements

This work is funded by the Federal Ministry of Education and Research (BMBF Germany, No. 01IS23007B).

6 REFERENCES

[Bay06] Bay, H., Tuytelaars, T., and Van Gool, L. SURF: Speeded Up Robust Features. European Conference on Computer Vision, 2006.

[Boo89] Bookstein, F. Principal Warps: Thin-Plate Splines and the Decomposition of Deformations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989.

[Bro92] Brown, L.G. A Survey of Image Registration Techniques. ACM Computing Surveys, 24(4), 325-376, 1992.

[Cha83] Chan, T.F., Golub, G.H., and LeVeque, R.J. Algorithms for computing the sample variance: Analysis and recommendations. The American Statistician, vol.37, no.3, pp.242-247, 1983.

[Che22] Chen, H., Luo, Z., Zhou, L., Tian, Y., Zhen, M., Fang, T., McKinnon, D., Tsin, Y., and Quan, L. ASpanFormer: Detector-Free Image Matching with Adaptive Span Transformer. European Conference on Computer Vision, 20-36, 2022.

[Eva08] Evangelidis, G.D., and Psarakis, E.Z. Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(10), 1858-1865, 2008.

[Hua22] Huang, Z., Ma, X., Wang, Y., Zhang, Y., Xu, Y., and He, Y. PDC-Net: Learning Patchwise Deformable Offsets for Dense Correspondence. European Conference on Computer Vision, 243-261, 2022.

[Joh16] Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. European Conference on Computer Vision, 694-711, 2016.

[Kan18] Kanazawa, A., Tulsiani, S., Efros, A.A., and Malik, J. Learning Category-Specific Mesh Reconstruction from Image Collections. European Conference on Computer Vision, 371-386, 2018.

[Lin23] Lindenberger, P., Sarlin, P.-E., and Pollefeys, M. LightGlue: Local Feature Matching at Light Speed. Proceedings of the International Conference on Computer Vision, 17627-17638, 2023.

[Mik05] Mikolajczyk, K., Tuytelaars, T., Schmid, C. et al. A Comparison of Affine Region Detectors. Int J Comput Vision 65, 43-72 (2005).

[Ngu18] Nguyen, T., Lalonde, J., and Gagnon, G. Deep Image Homography Estimation for Dynamic Scenes. IEEE Transactions on Image Processing, 27(11), 5272-5284, 2018.

[Sch17] Schops, T., Schonberger, J.L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., and Geiger, A. A multi-view stereo benchmark with high-resolution images and multi-camera videos. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.3260-3269, 2017.

[Low04] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60(2), 91-110, 2004.

[Tan22] Tan, D., Liu, J.-J., Chen, X., Chen, C., Zhang, R., Shen, Y., Ding, S., and Ji, R. Eco-tr: Efficient correspondences finding via coarse-to-fine refinement. European Conference on Computer Vision, pp.317-334, 2022.

[Che22] Chen, H., Luo, Z., Zhou, L., Tian, Y., Zhen, M., Fang, T., Mckinnon, D., Tsin, Y., and Quan, L. Aspanformer: Detector-free image matching with adaptive span transformer. European Conference on Computer Vision, pp.20-36, 2022.

[Rub11] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. ORB: An Efficient Alternative to SIFT or SURF. IEEE International Conference on Computer Vision, 2564-2571, 2011.

[Sar20] Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. SuperGlue: Learning Feature Matching with Graph Neural Networks. Proceedings of the Conference on Computer Vision and Pattern Recognition, 4938-4947, 2020.

[Sor07] Sorkine, O., Alexa, M., and Cohen-Or, D. As-Rigid-As-Possible Shape Deformation. ACM Transactions on Graphics, 26(3), 30, 2007.

[Sun21] Sun, J., Shen, Z., Wang, Y., Bao, H., and Zhou, X. LoFTR: Detector-free local feature matching with transformers. Proceedings of the Conference on Computer Vision and Pattern Recognition, pp.8922-8931, 2021.

[Syk09] Sýkora, D., Dingliana, J., and Collins, S. As-rigid-as-possible image registration for hand-drawn cartoon animations. Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering, pp.25-33, 2009.

[Sei23] Seibt, S., Kuth, B., von Rymon Lipinski, B., Chang, T., and Latoschik, M.E. Multidimensional image morphing—Fast image-based rendering of open 3D and VR environments. Virtual Reality and Intelligent Hardware (VRIH) Journal, Article Reference (Elsevier): VRIH195, 2023.

[Wan24] Wang, T., Huang, H., Cai, Z., Song, J., and Yang, J. 360°panorama stitching method with depth information: Enhancing image quality and stitching accuracy. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol.48, pp.191-197, 2024.

[Hua24] Huangfu, Y., Chen, H., Huang, Z., Li, W., Shi, J., and Yang, L. Research on a panoramic image stitching method for images of corn ears, based on video streaming. Agronomy, vol.14, no.12, p.2884, 2024.

[Dar24] Darzi, F., and Bocklitz, T. A review of medical image registration for different modalities. Bioengineering, vol.11, no.8, p.786, 2024.

[Wan24] Wang, H., Ni, D., and Wang, Y. Recursive deformable pyramid network for unsupervised medical image registration. IEEE Transactions on Medical Imaging, 2024.

[Wan04] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612, 2004.

[Zha21] Zhao, J., Liu, S., Peng, X., and Lin, X. Learning Invariant Representations for Unsupervised Image Registration. IEEE Transactions on Image Processing, 30, 2021.

[Zha18] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. Proceedings of the IEEE conference on computer vision and pattern recognition, 586-595, 2018.