Deep Lucas-Kanade Homography for Multimodal Image Alignment

Yiming Zhao    Xinming Huang    Ziming Zhang
Worcester Polytechnic Institute
100 Institute Rd, Worcester, MA, USA
{yzhao7, xhuang, zzhang15}@wpi.edu

Abstract

Estimating homography to align image pairs captured by different sensors or image pairs with large appearance changes is an important and general challenge for many computer vision applications. In contrast to others, we propose a generic solution to pixel-wise align multimodal image pairs by extending the traditional Lucas-Kanade algorithm with networks. The key contribution in our method is how we construct feature maps, named as deep Lucas-Kanade feature map (DLKFM). The learned DLKFM can spontaneously recognize invariant features under various appearance-changing conditions. It also has two nice properties for the Lucas-Kanade algorithm: (1) The template feature map keeps brightness consistency with the input feature map, thus the color difference is very small while they are well-aligned. (2) The Lucas-Kanade objective function built on DLKFM has a smooth landscape around ground truth homography parameters, so the iterative solution of the Lucas-Kanade can easily converge to the ground truth. With those properties, directly updating the Lucas-Kanade algorithm on our feature maps will precisely align image pairs with large appearance changes. We share the datasets, code, and demo video online.

1. Introduction

Pixel-wise alignment of multimodal image pair is an important problem for many computer vision applications in the field of medical imaging [12], remote sensing [45], and robotics [5]. For example, in the task of GPS denied navigation [16], the drone needs to align the RGB image from the camera with the existing map from the satellite. Aligning those two images suffers from multimodal color representations as well as appearance changes. In Fig. 1, we show a cross-season image pair with appearance changes and a Google Map and Satellite image pair with multimodal representations. To solve this problem, we propose a generic pipeline by extending the traditional Lucas-Kanade (LK) algorithm with neural networks. The key component is how to extract the feature map, named as deep Lucas-Kanade feature map (DLKFM). DLKFM is able to spontaneously recognize invariant features in various multimodal cases, like the roads and houses in the cross-season example or map labels in the Google Map and Satellite example. DLKFM also helps the LK to converge successfully by shaping the landscape of the objective function.

Homography or perspective transform is a general 2D image transformation, which maps pixel from one image to the other, denoted as \( \hat{x} = \hat{H} \hat{x}' \), where \( \hat{x} \) and \( \hat{x}' \) are homogeneous coordinates, \( \hat{H} \) is an arbitrary \( 3 \times 3 \) matrix. However, in 3D computer vision, parallax arises due to the change of the viewpoint, thus creates ambiguity for aligning 2D images. In this paper, we consider the same flat world assumption as [16], that the depth disparity is negligible like some UAV and remote sensing applications [32, 46]. Then, aligning the image pair is equivalent to estimating the ho-
mography matrix.

To estimate the homography, a straightforward solution is finding more than four matched points. This idea can be broadly summarized as the feature-based method including traditional SIFT [27], SURF [3], or recent deep learning methods D2 [13], LF-Net [34], R2D2 [37], etc. To further solve the challenge on multimodal images, some special features are also designed [45, 44]. However, those feature-based methods are facing the same generalizability problem. Even for those learning-based features, ground truth labels provided by 3D reconstruction systems are necessary [38, 39]. This prohibits the generic training of those feature detectors and descriptors on multimodal images.

Besides feature-based methods, direct method is the other way to calculate homography [2]. As the most famous one, the Lucas-Kanade family [28] has a long history and still stimulates new ideas in different aspects of computer vision [35, 43]. In this paper, we build a generic multimodal image alignment pipeline by extending Inverse Compositional Lucas-Kanade (IC-LK).

**Inverse Compositional Lucas-Kanade Algorithm.** Let $X_I$ and $X_T$ represent input image and template image, respectively, the Lucas-Kanade objective is stated as follows

$$\min_P ||X_T - W(X_I, P)||_2^2 \tag{1}$$

where $W(\cdot|P)$ is the warping function which warps the input image based on transformation parameter set $P$.

The Lucas-Kanade  algorithm iteratively solves for the warp parameters $P_{k+1} = P_k + \Delta P$. At every iteration $k$, the warp increment $\Delta P$ is obtained by linearizing

$$\min_{\Delta P} ||X_T - W(X_I, P_k + \Delta P)||_2^2 \tag{2}$$

using first order Taylor expansion

$$\min_{\Delta P} ||X_T - W(X_I, P_k) - \frac{\partial W(X_I, P_k)}{\partial P} \Delta P||_2^2 \tag{3}$$

Here, $\partial W(X_I, P_k)/\partial P$ needs to be recomputed at every iteration as it depends on $W(X_I, P_k)$. The inverse compositional(IC) [2] avoids this recomputing by applying the warp increments $\Delta P$ to the template instead of the input

$$\min_{\Delta P} ||W(X_T, \Delta P) - W(X_I, P_k)||_2^2 \tag{4}$$

using the warp parameter update $P_{k+1} = P_k + (\Delta P)^{-1}$ as where $\Delta P$ is the inverse increment mapping. In the corresponding linearized equation

$$\min_{\Delta P} ||X_T + \frac{\partial W(X_T, 0)}{\partial P} \Delta P - W(X_I, P_k)||_2^2 \tag{5}$$

$\partial W(X_T, 0)/\partial P$ does not depend on $P_k$ and can thus be pre-computed, resulting in a more efficient algorithm.

**Limitation of IC-LK on Multimodal Images.** IC-LK keeps accumulating $\Delta P$ toward ground truth, thus it usually has an accurate performance if it can work successfully. However, Eq. 1 assumes the color difference between the input image and template image achieves the minimum value once they are aligned. This brightness consistency assumption does not hold when aligning multimodal image pairs as the input and template usually have different color pattern representations. Furthermore, minimizing Eq. 1 is a non-linear optimization task as the input image depends non-linearly on the warp parameters $P$, so there is no guarantee the first-order approximation solution of Eq. 5 can have a $\Delta P$ moving along the right direction to the ground truth without getting into the local minimum.

**Our Solution.** To overcome the limitation of IC-LK on multimodal images, we hope feature maps processed after a Siamese network can satisfy those assumptions of IC-LK. To achieve this, we design a feature constructor inspired by classical feature extractors, such as Harris Corner [9]. This feature constructor constructs one channel feature map by using eigenvalues of the local covariance matrix on output tensor. Then, we design a special loss function with two terms. The first term helps feature maps to satisfy the brightness consistency assumption. The second term shapes the landscape of the Lucas-Kanade objective function, to help the first-order linearized solution $\Delta P$ can successfully keep moving toward the ground truth homography parameters. We show examples of learned deep Lucas-Kanade feature map (DLKFM) in Fig. 1.

**Contributions.** In general, we extend the well-known traditional Lucas-Kanade method with neural networks. In regard to applications [16, 46, 31], we propose a generic pipeline to align multimodal images. In regard to technical contributions, we develop two new specific technologies:

- **Feature constructor on multi-channel tensors.** In order to avoid updating homography on heavy tensors, the single-channel DLKFM is constructed on the high dimensional network output. We calculate the covariance matrix of vectors in each $3 \times 3$ patch. Then, we approximate the ratio between the largest eigenvalue and the trace as an indicator to form a single channel feature map. In the ablation study section, we show this operation improves the performance by extracting meaningful features on the single-channel feature map.

- **Special loss function for the convergence of Lucas-Kanade.** We design a special loss function based on constructed feature maps. This loss function makes two feature maps satisfy brightness consistency. It further helps the convergence of the Inverse Compositional Lucas-Kanade algorithm by shaping the optimization objective with a supportive convex function. The landscape of the objective function on learned feature maps is visualized in the section of ablation study.
2. Related Work

Image alignment is one of the most fundamental computer vision tasks with a long history [2, 41, 16]. Direct methods [28], feature-based methods [25] and their combinations [1] are traditional ways to align images. Those non-learning methods often assume input and template images have the same texture pattern for the same feature, which is not the case for multimodal images. To overcome this, some researchers try to design special feature detectors [45, 44], which are usually hard to prove the generalizability.

With the coming of the deep learning epoch, people are curious if the deep neural network can learn the corresponding features on two images [26, 8]. The positive answer attracts many recent research papers investigating how to utilize the strong ability of neural networks for image matching, like semantic matching or finding dense correspondences [19, 20, 42]. Extending feature detectors and descriptors with the neural network is also a hot topic. Recent models, like D2 [13], LF-Net [34], R2D2 [37], RF-Net [40], Elf [4] are all showing strong ability to handle struggling cases for traditional feature detectors. However, many of those methods rely on ground truth feature points to train the network. Those ground truth feature points are traditional features extracted by 3D reconstruction systems, like Colmap [38, 39]. This limits the training of those methods on multimodal datasets.

A more straightforward idea is making use of the strong approximation ability of networks to directly regress homography parameters [10, 14]. Those deep homography papers usually focus on specific challenges, like how to design the unsupervised loss [32], or how to handle dynamic scenes [23, 47]. Although using networks to do regression is simple, it always can work under various conditions. However, the model accuracy is usually not guaranteed.

In contrast to above methods, we extend the traditional Inverse Compositional Lucas-Kanade method on multimodal image pairs. Two recent papers explore Lucas-Kanade with neural networks [6, 29]. Authors of one paper replace some parts of the inverse compositional solution with trainable data-driven network modules [29]. However, their method mainly focuses on solving the 3D rigid motion estimation problem. In CLKN [6], authors design a loss function to guarantee the first updating always bring relative large progress, the further iterative updating is conducted on feature maps. Since there is no special treatment for feature maps, they imply the same color pattern will become the same feature via same network parameters. This assumption does not work for multimodal image pairs as two images have different color textures for the same place.

3. Method

3.1. Problem Settings

To overcome the limitation of Lucas-Kanade on multimodal image pairs, we extend the Inverse Compositional Lucas-Kanade on feature maps. A Siamese network, which has one template branch defined as $F(\cdot|w_T)$, and one input branch defined as $G(\cdot|w_I)$, is used to extract feature maps. Then, the objective function in Eq. 1 can be generalized as:

$$\min_{\mathbf{P}} \|F(X_T, \bar{w}_I) - W(G(X_I, \bar{w}_I), \mathbf{P})\|^2_2.$$  (6)

Here, we have two kinds of set of parameters, the Siamese network parameter set $\{w_T, w_I\}$, and homography parameter set $\mathbf{P}$. During training, we use ground truth $\bar{\mathbf{P}}$ to train $\{w_T, w_I\}$, then the fixed network weights $\{\bar{w}_T, \bar{w}_I\}$ are used to update $\mathbf{P}$ during inference to minimize Eq. 6. The iteration inverse compositional solution of Eq. 6 is

$$\Delta \mathbf{P} = (J^T J)^{-1} J^T r,$$  (7)

where

$$J = \nabla F(X_T, \bar{w}_T) \cdot \frac{\partial W}{\partial \mathbf{P}}(F(X_T, \bar{w}_T), 0),$$  (8)

$$r = \left[ \begin{array}{c} W(G(x_I, \bar{w}_I), P)_1 - F(X_T, \bar{w}_T)_1 \\ \vdots \\ W(G(x_I, \bar{w}_I), P)_N - F(X_T, \bar{w}_T)_N \end{array} \right].$$  (9)

The derive details from Eq. 6 to Eq. 7 can be found from equation (2) to (8) here [6]. Since the updating of Eq. 7 is conducted on trained feature maps during inference, we use $\{\bar{w}_T, \bar{w}_I\}$ to represent fixed network weights during inference. In Eq. 9, $N$ is the total number of elements on the feature maps.

3.2. Construct Single Channel Feature Map

The modern convolution layers usually output a high dimension tensor with multiple filters, like 64, 128, or more. This involves a large computational burden to update $\Delta \mathbf{P}$ in Eq. 7 and Eq. 9. Some recent papers, which also try to involve neural networks into the pipeline of Lucas-Kanade, usually take straightforward choices like reducing the number of filters in the last layer to a small value [6], or applying channel-wise summation on the output tensor [29]. In contrast to them, we construct a single channel feature map based on eigenvalues of the local covariance matrix.

Considering a large tensor $B$, we apply a $3 \times 3$ sliding window to extract 9 vectors locally. Then, we calculate the covariance matrix of those 9 vectors. We use $B_{i,j}$ to denote the covariance matrix calculated by the local patch centered...
at \((i, j)\) of tensor \(B\). Intuitively, if the information captured by this local patch is almost the same, those 9 vectors should be similar. Thus, the covariance matrix will not have a dominant eigenvalue. So we construct a one channel feature map \(C\) by using the ratio of the largest eigenvalue over the summation of all eigenvalues:

\[
C_{i,j} = \frac{\lambda_{\text{largest}}}{\text{Trace}(B_{i,j})},
\]

where \(\lambda_{\text{largest}}\) is the largest eigenvalue of \(B_{i,j}\). In traditional image processing, we as humans think corners or edges are important features for an image, like the Harris corner detector [9] or Canny edge detector [11]. In many different subjects which need matrix analysis, like statistics and signal processing, the ratio of largest eigenvalue over trace plays an important role [30]. So we think this ratio is an important feature for high dimensional tensor, just like the edge is an important feature for the image.

However, we found calculating and doing backpropagation on the largest eigenvalue for each patch is super slow with modern deep learning frameworks. Then, we design a more practical indicator by following Perron–Frobenius theorem [36, 15]. According to Perron–Frobenius theorem, the largest eigenvalue of the covariance matrix has an upper bound which is the maximum of summation of each row, and a lower bound which is the minimum of summation of each row. So the new feature map of a high dimension tensor is constructed by:

\[
C_{i,j} = \frac{\max_m(\sum_n B_{i,j}^{m,n}) + \min_m(\sum_n B_{i,j}^{m,n})}{2 \times \text{Trace}(B_{i,j})},
\]

where \(m, n\) are row and column numbers of the covariance matrix. The calculation of this feature map is just like doing regular convolution, which keeps our method practical for training and testing. We discuss more details in the section of the ablation study.

3.3. Build Loss Function

Brightness Consistency Loss Term. Once we have two feature maps from the template branch and the input branch in the Siamese network, we need to design a loss function to help those two feature maps learn meaningful features for Eq. 7. The original Lucas-Kanade objective Eq. 1 assumes that the color difference should be the minimum once two images are well aligned. We apply this assumption on feature maps to get the first term of our loss function named Brightness Consistency Loss \(l_{bc} = E(w_T, w_I|\hat{P})\), where

\[
E(w_T, w_I|\hat{P}) = \|F(X_T, w_T) - W(G(X_I, w_I), \hat{P})\|_2^2
\]

(12)

Compared with Eq. 6, \(l_{bc}\) has trainable \(\{w_T, w_I\}\) with fixed ground truth \(\hat{P}\) on training set. The warping function is implemented by using STN [18] for backpropagation.

Convergence Loss Term. Original objective Eq. 1 is a highly non-linear function, whereas the iteration solution Eq. 7 relies on gradient descent approximation. To help it converge better in the right direction, we hope the objective Eq. 6 on feature maps can achieve local minimum at ground truth \(\hat{P}\) and have a smooth surface around \(\hat{P}\) without the other minimum. Control the convexity of a function usually needs to explicitly calculate Jacobian or Hessian, but Eq. 6 is a multivariable function of \(P\) with a Siamese network as the backbone. Instead of directly calculating Jacobian or Hessian, we consider a supportive convex function \(g(Z)\) with \(J(g(Z))|_{Z=\hat{P}} = 0\) to shape the local region of objective function around \(\hat{P}\). The \(Z\) is a vector with eight parameters and \(g(Z)\) has the minimum value at \(Z = \hat{P}\). We push the optimization objective function Eq. 6 satisfy two conditions in a small region \(\Theta\) around \(\hat{P}\) by training:

\[
\text{Condition 1}: \forall (\hat{P} + \Delta P) \in \Theta, \quad E(\hat{P} + \Delta P|w_T, w_I) - E(\hat{P}|w_T, w_I) \geq g(\hat{P} + \Delta P) - g(\hat{P}),
\]

This condition guarantees that Eq. 6 achieves local minimum at \(\hat{P}\) with a relatively larger local change than \(g(Z)\).

\[
\text{Condition 2}: \forall P \in \Theta, \nabla_v E(P|w_T, w_I) \geq \nabla_v g(P),
\]

where \(\nabla_v\) is the directional derivative. Since Condition 1 is about the single point at \(\hat{P}\), Condition 2 guarantees that Eq. 6 has a steeper directional derivative than \(g(Z)\) in a local region around \(\hat{P}\). Compared with partial derivative, directional derivative is easier to be implemented for function without explicit expression.

In our implementation, we use the simplest \(g(Z) = \sum_{i=1}^{8} (Z_i - \hat{P}_i)^2\) as the supportive function. Then, the loss term for Condition 1 can be defined as:

\[
l_{con1} = -\frac{1}{M} \sum_{m=1}^{M} \min(0, E(\hat{P} + \Delta P_m|w_T, w_I) - E(\hat{P}|w_T, w_I) - \sum_{i=1}^{8} (\Delta P_i^m)^2),
\]

(13)

where \(\Delta P_i^m\) is a sampled random noise of the \(i\)-th variable in vector \(\hat{P}\), we sample \(M\) times for each training batch.

For Condition 2, we define \(\forall (\hat{P} + \Delta P) \in \Theta, P = \hat{P} + \lambda \Delta P\), where \(\lambda \in (0, 1)\). Then, we have:

\[
\nabla_\Delta P E(P|w_T, w_I) = \lim_{(1-\lambda) \to 0} \frac{E(\hat{P} + \Delta P|w_T, w_I) - E(\hat{P} + \lambda \Delta P|w_T, w_I)}{(1-\lambda)\|\Delta P\|},
\]

(14)

\[
\nabla_\Delta P g(P) = \lim_{(1-\lambda) \to 0} \frac{\Delta P^2 - (\lambda \Delta P)^2}{(1-\lambda)\|\Delta P\|},
\]

(15)
Combining Eq. 14 and Eq. 15, we have the mathematical expression of Condition 2 as
\[
\forall (\hat{P} + \Delta P) \in \Theta, E(\hat{P} + \Delta P | \bar{w}_T, \bar{w}_T) \\
- E(\hat{P} + \lambda \Delta P | \bar{w}_T, \bar{w}_T) \geq (1 - \lambda^2) \sum_{i=1}^{8} (\Delta P_i^m)^2. \tag{16}
\]

The loss term for Condition 2 can be defined as:
\[
l_{con2} = - \frac{1}{M} \sum_{m=1}^{M} \text{minimum}(0, E(\hat{P} + \Delta P_m | w_T, w_T) \\
- E(\hat{P} + \lambda \Delta P_m | w_T, w_T)) - (1 - \lambda^2) \sum_{i=1}^{8} (\Delta P_i^m)^2).
\tag{17}
\]

Since we have \( \lambda \in (0, 1) \) and \((1 - \lambda) \to 0\), we choose \( \lambda = 0.8 \) for the implementation.

Finally, the Convergence Loss is defined as \( l_{converge} = l_{con1} + l_{con2} \). The total loss function is the summation of the Brightness Consistency Loss and the Convergence Loss
\[
l_{total} = l_{bc} + \gamma l_{converge} \tag{18}
\]
where \( \gamma \) is a hyperparameter to control the balance. We choose \( \gamma = 0.1 \) in this paper.

### 3.4. Network Structure

We show the network structure in Fig. 2. The Siamese network has three identical blocks except for the first layer. In each block, there are 8 convolution layers with 64 filters connected with the residual. The first layers in the second and third blocks have stride 2 to downsample tensors. All the other layers have \( 3 \times 3 \) kernels with stride 1.

During training, the input image and template image go through two branches. Then, we extract output tensors after each block, and apply Eq. 11 to construct feature maps for both input and template. The loss function Eq. 18 is defined based on constructed feature maps in three different scales.

During inference, we have feature maps in three different scales. The Eq. 7 is updated bottom-up from coarse to fine. We give an example of the updating during inference in Fig. 3. Note that the iterative solution needs an initial guess to start with and a stopping criterion to end, we discuss those two in the section of experiments.

### 4. Experiments

#### 4.1. Datasets

To test the performance of our method, we prepare three datasets. We utilize Adam [21] as our optimizer with a learning rate 0.0001 to train 10 epochs for all three datasets. As discussed in the section of method, we set \( \gamma = 0.1 \), \( \lambda = 0.8 \) and \( M = 4 \). All the experiments are conducted on a single RTX 2080 Ti.

**MSCOCO** [24] is a widely used large scale dataset. Recent deep homography papers also choose this dataset as a test bench. We exactly follow the settings of those papers [10, 6, 23]. The whole training set of MSCOCO is used for our training, and we sample 6K images from the validation set as our test. Each sample is resized to \( 196 \times 196 \) as an input image. Then, we randomly choose four points in four \( 64 \times 64 \) boxes at the corner and warp the chosen region to \( 128 \times 128 \) as the template. An example is showed in Fig. 4.

The use of this regular dataset is serving for two purposes. First, since we implement several baseline methods, including DHN [10], MHN [23], SIFT with Ransac, etc, observing the performance drop from the same model on multimodal datasets will illustrate the challenge of estimating homography on multimodal images. Second, we want to show our method can also work on the regular image alignment task.
**4.2. Evaluation Metric**

The same as recent papers [6, 23], we are using average corner error in pixel as the evaluation metric:

\[
e_c(P, \bar{P}) = \frac{1}{4} \sum_{j=1}^{4} ||W(e_j, P) - W(e_j, \bar{P})||_2,
\]

where \(e_1, ..., e_4\) are four corners in template image. The corner error measures the \(L_2\) distance of the warped corners and then takes the average over the four corners.

**4.3. Initial Homography**

Updating Eq. 7 on DLKFM needs an initial homography \(P_0\). The initial used in Fig. 3 is a pre-assigned homography matrix that we simply set \((31,31), (31,159), (159,159), (159,31)\) as four corners of a 128 × 128 template image on 192 × 192 input. Note, the closer the initial homography to the ground truth, the larger chance Lucas-Kanade will converge to it. In some applications, like tracking [35] or navigation [7], the previous frame can serve as the initial candidate since the relative pose is small. Some traditional methods, like fast affine template matching [22], can also work to provide an initial guess with the brightness consistency property of our feature maps.

In this paper, we choose two recent network-based deep homography models, DHN [10] and MHN [23], to respectively provide the initial homography. This helps our solution achieve state-of-the-art performance on recent benchmark settings [6, 23]. It is also able to show the strong ability of our method to work as a post-process refiner that keeps improving the alignment accuracy based on existing or future solutions.

**4.4. Stopping Criterion**

The iterative solution Eq. 7 needs a stopping criterion during the inference. For each \(\Delta P\) on \(P_k\), the average four corner error can still be used to evaluate relative improvement of \(\Delta P\) as \(e_c(P_k + \Delta P, P_k)\). In this paper, we set the stopping criterion as \(e_c(P_k + \Delta P, P_k) < 1\) pixel for smallest scale, \(e_c(P_k + \Delta P, P_k) < 0.1\) pixel for middle scale, and...
$e_c(P_k + ΔP, P_k) < 0.01$ pixel for largest scale. To further control the inference time, the maximum iteration number on each scale is 30.

4.5. Model Performance

We show the performance of our method and all baseline methods in Fig. 5. In general, our solution outperforms the others on multimodal datasets with a large margin. We give further discussions about each method below.

**Vanilla.** Based on how we generate templates, the vanilla means centers of four corner boxes are used as the prediction. This is a baseline for guessing without knowledge.

**SIFT+RANSAC.** This is a typical traditional method to calculate homography. We use OpenCV implementation of SIFT, BF Matcher, and RANSAC. From Fig. 5, we can see it achieves pretty accurate performance on the MSCOCO dataset which has the exact same template and input images. However, on cross-season Google Earth dataset, it performs better than vanilla for only 60\% of images, and it completely fails to calculate homography on Google Maps.

**CLKN.** This method extends inverse compositional Lucas-Kanade on feature maps. Based on vanilla as initial, it relies on training to help the first updating get relatively large progress. However, since CLKN [6] does not set any constrain on feature maps, it fails to align multimodal images. We show the feature maps learned by CLKN on the MSCOCO dataset and the Google Earth cross-season dataset respectively in Fig. 6. It is clear to see learned feature maps on multimodal images are not meaningful.

**DHN [10] and MHN [23].** Those two methods are using the network as a regression function to directly output homography parameters. A recent technical report gives a comprehensive comparison between those two methods and many traditional methods under various conditions [33]. MHN [23] is a more advanced version as it combines three DHN [10] in a multi-scale cascaded way. Though the idea is straightforward, we find both two methods always can give meaningful results for different cases. However, compared with the regular MSCOCO dataset, we observe the performance drop of those two methods on multimodal datasets showed in Table 1. For example, MHN [23] performs much better than DHN [10], and aligns 95\% of total test images on MSCOCO within 3-pixel error, but it fails to align almost all test multimodal image pairs within 3 average pixels.

**DHN + DLKFM and MHN + DLKFM.** As our major contribution, DLKFM guarantees the local minimum and local smoothing of the Lucas-Kanade objective function. The iterative updating solution of Eq. 7 needs the initial homography to start with. As discussed before, we choose results from regression models DHN [10] and MHN [23] as the initial $P_0$ respectively. This makes our pipeline generic, which is not limited to the experimental settings in this paper. According to Fig. 5 and one recent technical report [33], those regression models are always able to give relatively good performance under different conditions. In Table. 1, we show the improvement by updating Eq. 7 on DLKFM. On multimodal datasets, our method largely improves the accuracy by helping most of the image pairs get smaller average corner errors than 3 pixels. On MSCOCO, even though...
5.1. Effectiveness of Feature Constructor

<table>
<thead>
<tr>
<th>Initial Method</th>
<th>Feature type</th>
<th>&lt;3 pixel</th>
<th>&lt;10 pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHN [10]</td>
<td>Regular Conv</td>
<td>47%</td>
<td>72%</td>
</tr>
<tr>
<td></td>
<td>DLKFM</td>
<td>60%</td>
<td>83%</td>
</tr>
<tr>
<td>MHN [23]</td>
<td>Regular Conv</td>
<td>55%</td>
<td>83%</td>
</tr>
<tr>
<td></td>
<td>DLKFM</td>
<td>66%</td>
<td>91%</td>
</tr>
</tbody>
</table>

Table 2. Performance comparison on cross season Google Earth.

Since Eq. 8 and Eq. 9 need to be calculated with all values from the output tensor, we reduce the number of channels to one by using Eq. 11. The other common way is using a regular convolutional layer with fewer filters [6, 29]. We conduct an extra experiment on cross-season Google Earth dataset by replacing our feature constructor with a regular one filter convolutional layer. We list the comparison performance in Table 2. To further demonstrate the effectiveness of our feature constructor, we visualize the learned feature map in Fig. 7. It is clear that our feature constructor helps the feature map extract meaningful features, which are invariant roads and houses. In recent papers [46, 31], those features are learned with semantic labels, like the building label provided by OpenStreetMap [17], whereas our method spontaneously recognizes those patterns.

5.2. Effectiveness of Designed Loss Function

Our special designed loss function has two terms. The first term helps two feature maps keep the brightness consistency, which has been presented in Fig. 1. The second term helps the convergence of Eq. 6 by shaping the landscape of optimization function around the ground truth value \( \hat{P} \), that is \( \hat{P} = [\hat{P}_{11}, \hat{P}_{12}, \hat{P}_{13}, \hat{P}_{21}, \hat{P}_{22}, \hat{P}_{23}] \). To verify it, we keep six parameters fixed, and perturb the other two around the ground truth. On the image coordinate, we perturb scale \((\hat{P}_{11}, \hat{P}_{22})\) in two directions and translation \((\hat{P}_{13}, \hat{P}_{23})\) in two directions, shown in Fig. 8. The sample is from our multimodal Google Map and Satellite dataset. For original image pair, we can see the \( \Delta P \) has no chance to converge as the \( \hat{P} \) is even not the minimum. On the contrary, the landscape of the objective function on learned DLKFM shows a smooth surface with the minimum at \( \hat{P} \). Therefore, by minimizing the objective, \( \Delta P \) will keep moving toward the ground truth \( \hat{P} \).

6. Conclusion

This paper proposes a generic method to align multimodal images by extending Lucas-Kanade on feature maps. Solving this fundamental challenge has the potential to provide solutions for many computer vision applications in the filed like UAV or remote sensing. Two specific techniques have been developed as part of our method. Since how to extract meaningful features and how to shape the objective function are both widely existing challenges, we believe those two technical contributions can give a hint to those related computer vision or machine learning tasks.
References


