SELF-SUPERVISED HOMOGRAPHY ESTIMATION
(CMPUT 615 FINAL REPORT)

Mohammadjavad Matinkia
Department of Computing Science
University of Alberta
matinkia@ualberta.ca

April 26, 2022

ABSTRACT

Homography matrix plays a key role in multiview geometry and computer vision and has various applications such as image alignment, image rectification, and image registration. The classic methods for estimating the homography matrix between two views of a same object requires a prior knowledge on point correspondences as well as constraints such as brightness constancy. In this project, we propose a self-supervised/unsupervised deep learning method which leverages the power of high dimensional features to levitate the requirement for point correspondence and brightness constancy. We evaluate the proposed method on the Brick Motion data of the UCSB dataset and consider four different types of motions, namely panning, rotation, perspective transformation, and scaling, and we show that the proposed method demonstrates a reasonably good performance.

1 Introduction

Finding the transformation between two different views of a same object is a crucial task in computer vision and has several important applications such as image alignment, image rectification, and image registration. Depending on how the object is transformed, the transformation matrix between the two views can be categorized as Euclidean or rigid transformation, metric transformation, affine transformation, and projective transformation.

Each category of such transformations impose certain constraints on how the lines, points, and angles are mapped to each other. Euclidean (or rigid or isometry) transformation preserves the distances between the points and the angles between the lines. Therefore, a Euclidean transformation can be represented by a rotation and a translation. Metric transformations do not preserve the distances, rather they maintain the relative distances. The angles between the lines are kept intact in such transformations. Therefore, a metric transformation can be represented by a scaling factor, a rotation, and a translation. Affine transformations neither preserve the distances nor preserve the angles, rather they maintain the paralellism. Therefore, an affine transformation can be represented by a scaling factor, a shear, a rotation, and a translation.

The most general form of a transformation known as the projective transformation merely preserves the cross ratios, intersections, and tangencies. These transformations are represented by a 3×3 matrix with 8 degrees of freedom, known as the *homography matrix*. To be more precise, a homography matrix is represented as

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}, \tag{1}$$

where the entry h_{33} is usually set to 1 to show that the matrix has only 8 degrees of freedom. Applying the homography matrix to a point $\mathbf{x} \in \mathbb{R}^2$ requires that the point is represented in the homogeneous coordinates $\tilde{\mathbf{x}} \in \mathbb{P}^2$. Then, the transformed point can be obtained by

$$\mathbf{H}\tilde{\mathbf{x}} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2}$$

The traditional method for finding the homography matrix between two views of the same object requires to know at least four pairs of point correspondences. Knowing the point correspondences, the homography matrix is evaluated using the *Direct Linear Transformation (DLT)* algorithm. However, finding the point correspondences is a hard task itself and to find the point correspondences, usually, tracking methods are employed. Such tracking methods have their own limitations such as the brightness constancy constraint.

In this project we propose an unsupervised/self-supervised deep learning method to estimate the homography matrix between two views. The proposed method does not require point correspondences and does not rely on image brightness constancy. For two images $\mathbf{I}_a$ and $\mathbf{I}_b$, the model which is trained in an end-to-end manner, estimates the homography matrix between the two images, and warp the images to find the transformed images $\mathbf{I}'_a$ and $\mathbf{I}'_b$. Simultaneously, the model finds high dimensional representation $\mathbf{F}_a, \mathbf{F}_b$ for the images and high dimensional representations $\mathbf{F}'_a, \mathbf{F}'_b$. Using the extracted representation, we use a loss function based on the cosine similarity between the representations to train the model and find the homography matrix. We show that the proposed loss function shows better behaviour than the L_1-norm loss functions used in other related works. The main contributions of this project are as follows

- Proposing a novel model for homography estimation without the requirement for knowing the point correspondences,
- proposing a modified loss function with better representation learning behavior to train the model in an end-to-end manner.

The rest of this report is structured as follows; in Section 2 we review several recent works on deep homography estimation. In Section 3 we elaborate the method and the model. In Section 4 we illustrate some visual and quantitative results. In Section 5 we mention ideas for further research and improvement of the proposed method.

2 Related Works

The ever-increasing success of deep learning methods in numerous applications in computer vision, has driven the researchers to rethink the classical methods in computer vision and replace them with more efficient deep learning methods [1, 2, 3, 4]. Due to its importance in various applications, recently, researchers have concentrated on developing deep learning methods for homography matrix estimation. Here, we review some most recent attempts on unsupervised homography matrix estimation using deep neural networks.

Nguyen et. al. [5] propose an unsupervised model for approximating planar homographies. Even though their proposed model estimates the homography matrix using a VGG network and benefits from high dimensional features, the final loss function used for training the network is still solely dependent on the intensities of the images. Also, their model requires the initial four points corresponding to the corners of the object of interest in the image. Given the source image, target image, and the coordinates of the region of interest, their model estimates the 4-point version of the homography matrix. Further, they introduce a differentiable DLT algorithm suitable for deep neural architectures and exploit it to determine the homography matrix from its 4-point version. The final loss used for training the model is simply the mean difference between the intensities of the first image and the intensities of its warped version.

Zhang et. al. [6] also introduce an unsupervised deep model to estimate the homography matrix. Their proposed model, directly computes the homography matrix from two given input images. This work, which could be considered as the most similar to our project, extracts feature maps of the same size as the input images using a convolutional neural network. These feature maps are then fed to a mask prediction network which acts as an attention layer, attenuating the features that are not informative for the homography estimation task. The final feature maps are then stacked together and fed to a large ResNet34 network to produce the homography matrix. The loss function for training this model is defined as a triplet loss based on L_1-norm between the learned representations.

As one of the most recent attempts to estimate the homography between two views, Ye et. al. [7], propose to estimate a homography flow instead of a homography matrix. In essence, applying the homography matrix to each pixel of the image moves it in two directions. Hence, the result of applying the homography on the image can be regarded as a 2D flow over the image. Further, they introduce homography basis vectors and find the homography flow based on estimating the coefficients of the homography basis vectors.

As mentioned earlier, the most similar work to this project, is the unsupervised deep model proposed by [6]. However, it is noteworthy to mention that there are fundamental differences between this project and the work of [6]. The first

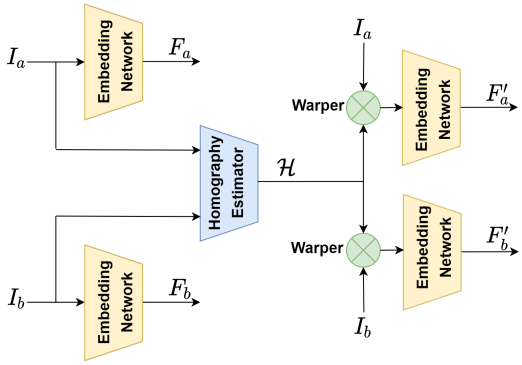

Figure 1: The schematic of the whole model containing the embedding networks and the homography estimation network. The whole model is trained in an end-to-end manner using the loss function of Eq.(5).

important difference is how we define the loss function to train the model. Here, we construct the loss function based on the cosine similarity criterion while in [6], the loss function is constructed upon the L_1-norm. The second difference is how we design the architecture of the networks and the whole model. Finally, the last difference is in the procedure by which the model is trained. In this project, the stacked images are directly fed to an embedding network to estimate the homographies, while in [6], the representations are learned first, and then, they are stacked together and used to estimate the homography matrix.

3 Proposed Method

First, we give an overview of the intuition of the proposed method. Besides the Direct Linear Transformation (DLT) algorithm to find the homography matrix which requires at least four pairs of point correspondences, a natural choice to find the homography matrix is to find the matrix $\mathbf{H}$ which minimizes the following mean squared error function

$$\mathcal{L}(\mathbf{I}_a, \mathbf{I}_b) = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{I_a}(\mathcal{W}(\mathbf{X_i}) - \mathbf{I}_b(X_i)\|, \tag{3}$$

where N denotes the total number of pixels and X_i is the coordinates of the i^{th} pixel. There are several notes in order regarding the above loss function. First, the norm in the expression of Eq.(3) is usually defined either as the l_1 or l_2 norms. Second, $\mathcal{W}(X_i)$ is the warp function which warps the pixel X_i using the homography matrix $\mathbf{H}$. One of the shortcomings of this loss function is that it relies on the image intensities, and therefore, it is constrained on the brightness constancy condition. An intuitive idea is to find the homography matrix using higher-dimensional feature maps. In other words, instead of using image intensities $\mathbf{I}_a, \mathbf{I}_b$, one can use high-dimensional representations $\mathbf{F}_a, \mathbf{F}_b$.

The overarching goal of this project is to estimate the homogrphy matrix using such high-dimensional representations. More precisely, the objective is to find a homography matrix such that the representation of warped image is as close as possible to the representation of the target image. One important difference between this project and other related works is that we use cosine similarity instead of any l_p norms to measure the closeness of the representations. Formally, to measure how close the representation $\mathbf{F}_a$ is to $\mathbf{F}_b$ we use cosine similarity as

$$\zeta_{ab} = \zeta(\mathbf{F}_a, \mathbf{F}_b) = \frac{\langle \mathbf{F}_a, \mathbf{F}_b \rangle}{\|\mathbf{F}_a\|_2 \|\mathbf{F}_b\|_2}, \tag{4}$$

where $\langle ., . \rangle$ denotes the inner product between two vectors. Cosine similarity is a widely used loss function in self-supervised settings, especially in contrastive learning. In section 4.3 we demonstrate how this closeness metric is better than the commonly used l_1 norm.

In this project, the high-dimensional representations are extracted using an *embedding network* which receives an image and generates high dimensional feature vectors. Also, the homography matrix is estimated using a *homography estimation network* which receives the concatenation of both images and estimates an 8 DOF homography matrix. The general process of estimating the homography matrix requires extracting the feature vector $\mathbf{F}_a$ of image $\mathbf{I}_a$, the feature vector $\mathbf{F}_b$ of $\mathbf{I}_b$, the feature vector $\mathbf{F}'_a$ of $\mathbf{I}_a(\mathcal{W}(\mathbf{H}_{ab}, X))$, and the feature vector $\mathbf{F}'_b$ of $\mathbf{I}_b(\mathcal{W}(\mathbf{H}_{ba}, X))$. This requirements are obtained as follows:

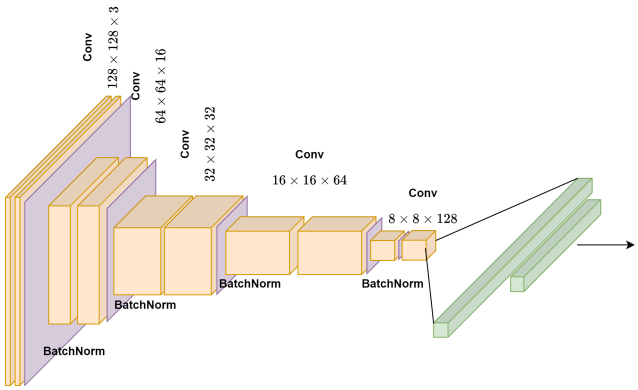

Figure 2: The neural architecture used for both embedding network and the homograpjy estimation network. The architecture contains 10 convolutional layers, each of two convolutional layers are followed by a batch normalization layer and the output of the final convolutional layer is flattened and fed to a two-layer fully-connected network to generate the representations or the homography matrix.

1. Using $\mathbf{I}_a, \mathbf{I}_b$, the representations $\mathbf{F}_a, \mathbf{F}_b$ are extracted using an embedding network.
2. $[\mathbf{I}_a; \mathbf{I}_b]$ ($[.;.]$ denotes the concatenation operator) is fed to the homography estimation network to estimate $\mathbf{H}_{ab}$; similarly $[\mathbf{I}_b; \mathbf{I}_a]$ is fed to the same homography estimation network to estimate $\mathbf{H}_{ba}$.
3. $\mathbf{I}_a(\mathcal{W}(\mathbf{H}_{ab}, X))$ and $\mathbf{I}_b(\mathcal{W}(\mathbf{H}_{ba}, X))$ are generated.
4. $\mathbf{I}_a(\mathcal{W}(\mathbf{H}_{ab}, X))$, $\mathbf{I}_b(\mathcal{W}(\mathbf{H}_{ba}, X))$ are fed to the embedding networks to generate the representations $\mathbf{F}'_a, \mathbf{F}'_b$.
5. The whole model is trained in an end-to-end manner using the following loss function:

$$\mathcal{L}(\mathbf{H}; \mathbf{I}_a, \mathbf{I}_b) = \zeta_{a'b} + \zeta_{b'a} - \lambda(\zeta_{ab} + \zeta_{a'b'}) + \mu\|\mathbf{H}_{ab}\mathbf{H}_{ba} - \mathbf{I}\|_2^2, \tag{5}$$

where $\zeta_{a'b} = \zeta(\mathbf{F}'_a, \mathbf{F}_b)$. $\zeta_{b'a}$ is defined similarly. Also ζ_{ab} and ζ_{ba} are defined as in Eq.(4).

The schematic of the whole model is illustrated in Figure (1). From a practical perspective, the upper two embedding networks share their parameters and they are essentially a same network. The similar is true for the lower two embedding networks. The network architecture for the embedding network and the homograophy estimation network is the same and is illustrated in Figure (2). In this architecture there are 10 convolutional layers and after every two convolutional layer a batch normalization layer is placed to stabilize the training procedure and avoid gradient vanishing and explosion. The final layer of the convolutional network is flattened and fed to a two=layer fully connected network to generate the representations. For the embedding network the output dimension indicates the dimension of the representations and for the homography estimation network the output dimension is fixed and is equal to 8, corresponding to the 8 degrees of freedom for the homography matrix.

3.1 Discussion on the Loss Function

Based on Eq.(5), the introduced loss function consists of five terms. Interpreting the functionality of these five terms is straightforward. The first two terms, $\zeta_{a'b}$ and $\zeta_{b'a}$ are intended to increase the similarity between the learned representations of the warped first image and the second image, and also increase the learned representations of the warped second image and the first image. In other words, we intend to find a homography matrix which leads to a same representation as the target image when applied to the first image and vice versa.

The second two terms, ζ_{ab} and $\zeta_{a'b'}$, are intended to decrease the similarity between the learned representations of both images and also decrease the similarity between the learned representations of warped images. These terms dictate the network to distinguish between the input images and their warped versions and avoid the optimizer to converge to a trivial solution of an identity matrix.

Finally, the last term, $\|\mathbf{H}_{ab}\mathbf{H}_{ba} - \mathbf{I}\|_2^2$,which stabilizes the structure of the learned homography matrix, drives the network to learn a homography matrix from the first image to the second image, such that the inverse homography matrix acts as the correct homography from the second image to the first image.

There are two regularization coefficients in the loss function of Eq.(5) which determine the importance of each term. Experimental analysis reveals that the proper value for λ is 0.01 and the proper value μ is 1, demonstrating the importance of a structure stabilizing term in the loss function.

4 Experimental Settings

In this section we evaluate the performance of the proposed model on several simple scenarios. We also provide an ablation study on the choice of the dimension of representations. We train the whole model for 2000 epochs and use the Adam optimizer to optimize the weights of the networks. The optimizer is set to have a learning rate of 10^{-4} and a weight decay parameter of 10^{-3}. The whole model is implemented in PyTorch and the code is available at here.

4.1 Dataset

As for the dataset, we use the Brick Motion data in the UCSB dataset [8]. This dataset contains several types of transformations from which we consider panning, rotation, perspective transformation, and scaling/zooming. The sample data used in this project is illustrated in Figure (3).

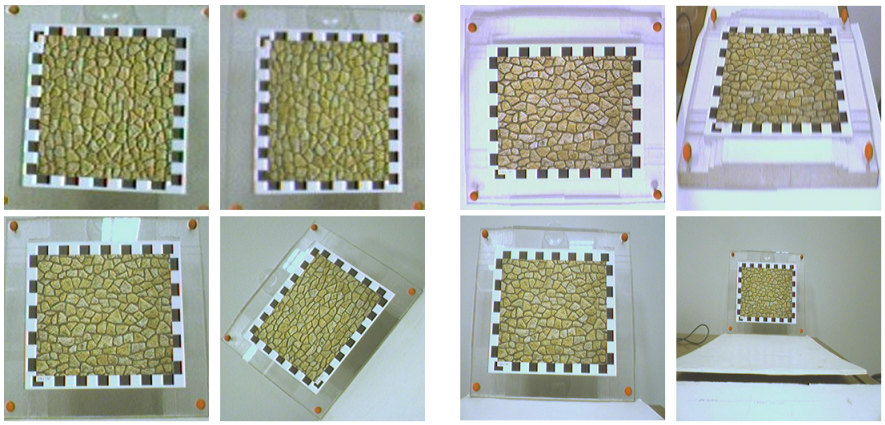

Figure 3: The data used for the project. In each block the left column depicts the first image and the second column depicts the second image or the target image.

In this figure, in each block, the left column shows the source image (or the first image or $\mathbf{I}_a$) and the second column shows the target image (or the second image or $\mathbf{I}_b$). The first row of the left block corresponds to the panning motion and the second row corresponds to a rotation. Also, the first row of the right block corresponds to a perspective transformation, and the second row corresponds to a scaling transformation.

4.2 Preprocessing

Before training the model with the input data, we take three preprocessing steps to have more fine-tuned results.

1. **Image resizing:** In this step we resize all input images to images of the size 128×128 pixels.
2. **Image normalization:** In this step we linearly normalize the images in each channel to the $[0, 1]$ interval according to

$$\mathbf{I}'_i = \frac{\mathbf{I} - \mathbf{I}_{\text{min}}}{\mathbf{I}_{\text{max}} - \mathbf{I}_{\text{min}}}, \quad i = 1, 2, \ldots, C, \tag{6}$$

   where C is the number of channels and is equal to 3 for RGB images. This type of image normalization stretches the intensities in the $[0, 1]$ interval by mapping the minimum intensity to 0 and maximum intensity to 1.

3. **Image centering:** As the last preprocessing step, we center the intensities in each channel around the mean intensity of that channel according to

$$\mathbf{I}'_i = \mathbf{I}'_i - \bar{\mathbf{I}}'_i, \quad i = 1, 2, \ldots, C. \tag{7}$$

4.3 Results

Before presenting the main results of the proposed model. It is worthwhile to briefly study the effect of representations dimensions on the training procedure and the final value of the loss function. To this end, we have plotted the training loss for four different values of the representations dimensions. Figure (4) illustrates the training loss for representation dimensions of $8, 16, 32$, and 64. As depicted in the figure, for all four values of the dimensions, the loss function finally converges, however, we can see that for 32 dimensions, the convergence is faster and more stable. Hence, throughout the experimenting phase the output dimension of the embedding networks are set to 32.

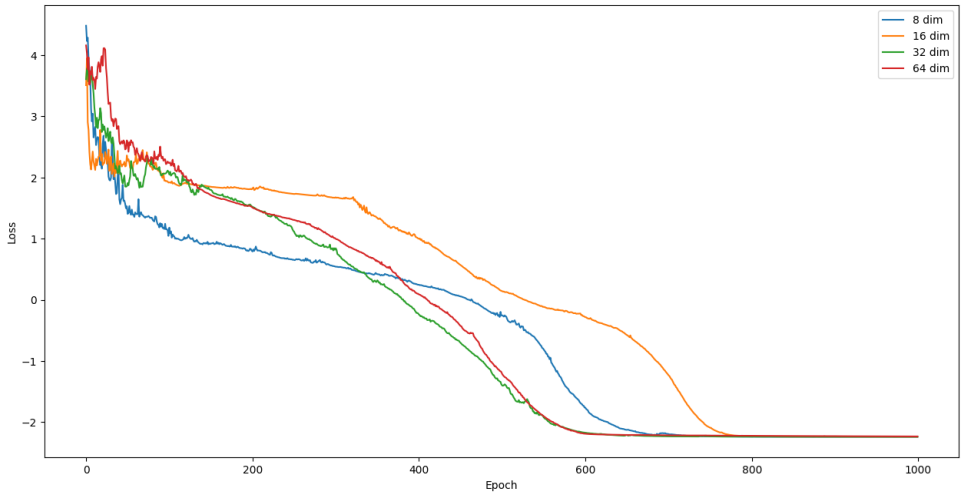

Figure 4: The training loss for different values of the representations dimensions. As it is shown in the diagram, for dimension 32, the learning curve converges faster and more stable.

Recall that as we introduced the loss function of Eq.(5), we mentioned that using the cosine similarity yields better, more informative, and more expressive representations. To understand this, consider Figure (5). In this figure, we have illustrated the learned representations in three dimensions after 1000 epochs. More precisely, the learned representations are transformed to 3 dimensions using PCA algorithm and are depicted here. The left plot show the representations when the cosine similarity is used, and the right plot shows the representations when the l_1 norm is used. In both plots, the red vectors denote the representations of the warped first image and the second image and the green vectors denote the representations of the warped second image and the first image.

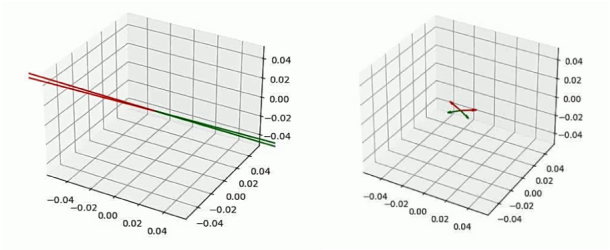

Figure 5: The illustration of the learned representations when cosine similarity is used (left), and when l_1 norm is used (right). We can see that in the case of cosine similarity, the desired similarity/dissimilarity behaviors are preserved more appropriately than when the l_1 norm is used.

As it is clear from the images, cosine similarity yields a better similarity/dissimilarity behavior among the representations comparing to l_1 norm. In fact, by looking at the loss function, we see that in the desired output, the red vectors should be placed as close as possible to each other, the green vectors should stay as close as possible to each other, while the pairs of red and green vectors should be as apart as possible from each other. This objective has been fulfilled desirably when we utilize the cosine similarity, while the l_1 norm fails to produce the representations with desired geometric configurations.

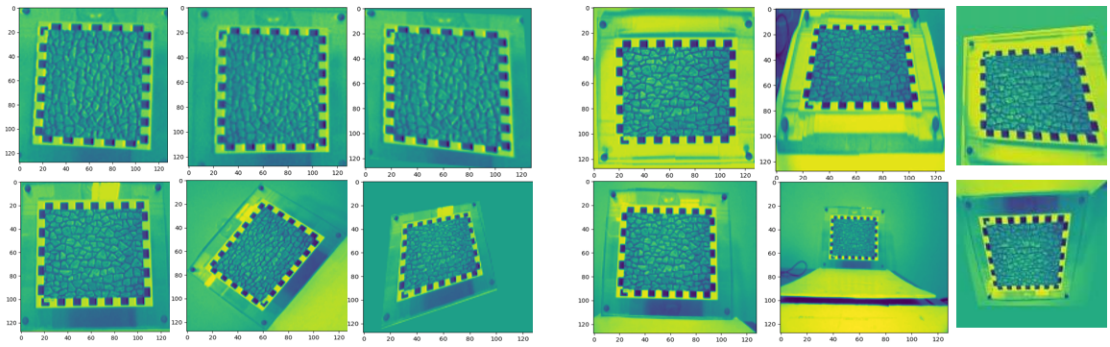

Figure 6: The illustration of the the result of estimated homographies. In each block, the left column shows the source image, the second column shows the second image, and the third column shows the result of applying the estimated homography to the first image.

As the final step, we demonstrate the visual performance of the model. To do this, we train the model for 1000 epochs and apply the estimated homography matrix on the first image. These results are depicted in Figure (6). In this figure, in each block, the left column shows the source image, the second column shows the second image, and the third column shows the result of applying the estimated homography to the first image.

5 Future Works

As it is shown in Figure (6), the final results could much improved. However, the results for a prototype model which uses light-weight networks are promising. Here, we propose several approaches that could be taken in order to improve the performance of the model. Similar to other works focusing on unsupervised deep homography estimation, one could employ more robust and more powerful networks such as ResNet, VGG, and Inception as the homography estimation network and the embedding networks and use the loss function of Eq.(5) to fine-tune the weights of the networks. However, there are other more theoretical threads that could be taken in order to improve the results. Empirical results demonstrate that the result of applying the homography matrix to an image is very sensitive to the values of the homography matrix entries. Hence, one could take approaches similar to variance-reduction techniques to bound and constraint the variance of the homography matrix entries. One other research thread could focus on extracting more informative representations by applying attention mechanism over the feature maps and feature vectors to emphasis on more useful features and attenuate unnecessary features. This could be promising especially because the current method does not take into account the initial position of the region of interest and extracts features from all over the image.

6 Conclusion

In this project we addressed one of the most fundamental problems in 3D computer vision, namely homography estimation, through a self-supervised/unsupervised setting. To this end, we proposed a deep neural model which estimates the homography matrix between two different views of a same object without requiring predefined point correspondences. We also proposed a novel loss function to train the model in an end-to-end manner. We showed that the proposed loss function has better representation learning behavior than the commonly used l_1 norm in other related works. Additionally, we visually demonstrated the performance of the proposed model over different types of transformations and showed that for a simple model the results are promising. Finally, we proposed several potential research threads that could enhance the performance of the model.

References

[1] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[2] Ping Hu, Bing Shuai, Jun Liu, and Gang Wang. Deep level sets for salient object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2300–2309, 2017.

[3] Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin. Deep learning on image denoising: An overview. *Neural Networks*, 131:251–275, 2020.

[4] Yiming Zhao, Xinming Huang, and Ziming Zhang. Deep lucas-kanade homography for multimodal image alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15950–15959, 2021.

[5] Ty Nguyen, Steven W Chen, Shreyas S Shivakumar, Camillo Jose Taylor, and Vijay Kumar. Unsupervised deep homography: A fast and robust homography estimation model. *IEEE Robotics and Automation Letters*, 3(3):2346–2353, 2018.

[6] Jirong Zhang, Chuan Wang, Shuaicheng Liu, Lanpeng Jia, Nianjin Ye, Jue Wang, Ji Zhou, and Jian Sun. Content-aware unsupervised deep homography estimation. In *European Conference on Computer Vision*, pages 653–669. Springer, 2020.

[7] Nianjin Ye, Chuan Wang, Haoqiang Fan, and Shuaicheng Liu. Motion basis learning for unsupervised deep homography estimation with subspace projection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13117–13125, 2021.

[8] Abhineet Singh and Martin Jagersand. Modular tracking framework: A fast library for high precision tracking. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3785–3790. IEEE, 2017.