ANKARA YILDIRIM BEYAZIT UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEEP LEARNING BASED HOMOGRAPHY PARAMETER ESTIMATION FOR MOSAICING OF NON-OVERLAPPING IMAGE PATCHES

M.Sc. Thesis by

Ali Batuhan YELLİ

Department of Electrical and Electronics Engineering

March, 2024

ANKARA

DEEP LEARNING BASED HOMOGRAPHY PARAMETER ESTIMATION FOR MOSAICING OF NON-OVERLAPPING IMAGE PATCHES

A Thesis Submitted to

The Graduate School of Natural and Applied Sciences of

Ankara Yıldırım Beyazıt University

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical and Electronics Engineering, Department of Electrical and Electronics Engineering

by

Ali Batuhan YELLİ

March, 2024

ANKARA

M.Sc. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**DEEP LEARNING BASED HOMOGRAPHY PARAMETER ESTIMATION FOR MOSAICING OF NON-OVERLAPPING IMAGE PATCHES**" completed by **Ali Batuhan YELLİ** under supervision of **Assist. Prof. Dr. Gökhan Koray GÜLTEKİN** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Gökhan Koray GÜLTEKİN

Supervisor

Prof. Dr. Ahmet KARAARSLAN

Jury Member

Assoc. Prof. Dr. Mustafa Mert ANKARALI

Jury Member

Prof. Dr. Sadettin ORHAN

Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All information, documents and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

Date: 2024, 6 February **Signature: ______________________________**

Name & Surname: Ali Batuhan YELLİ

ACKNOWLEDGEMENTS

First of all, I would like to express my endless gratitude to my advisor Assist. Prof. Dr. Gökhan Koray GÜLTEKİN. I would like to thank him for what he taught me and the support he gave me from the beginning of my master's degree to the completion of my thesis.

In addition to my advisor Assist. Prof. Dr. Gökhan Koray GÜLTEKİN, my family who was always with me during this difficult process, my school friends who I think we added something to each other by exchanging ideas in each other's studies, my undergraduate students I would like to thank my AYBU professors, who helped me write this thesis with what they taught me throughout my graduate education, and my colleagues and team leaders, who contributed to me both knowledgeably and morally throughout my business life.

March, 2024 **Ali Batuhan YELLİ**

DEEP LEARNING BASED HOMOGRAPHY PARAMETER ESTIMATION FOR MOSAICING OF NON-OVERLAPPING IMAGE PATCHES

ABSTRACT

Image mosaicing is the process of stitching two images with a larger field of view (FOV) to create a new large image with high resolution. Two kinds of approaches have generally been used for mosaicing. The first kind of approaches are called the traditional methods which are based on feature detection. Mosaicing quality depends on the number of features. Traditional methods do not perform well on images with low number of features. The supervised learning-based image mosaicing solutions unreliable because these methods often experience the problem of missing data. Studies in this field in the literature generally aim to stitch images with a common overlap area. In this article, we try to stitch images with translational displacements along the edges that do not have overlapping areas. For this, we propose a learning-based method and use a deep homography estimation network. The network is trained using a custom generated dataset to estimate the displacement amount of the target image in the horizontal and vertical axis relative to the reference image. Experimental tests are carried out using a custom generated dataset including landscape images and their gaussian noise added versions. The test results shows that the network can successfully predict the displacement amount in the horizontal and vertical axis between the two images and therefore the two images can be stitched from the right point.

Keywords: Computer vision, deep image mosaicing, deep homography estimation, image registration, image blending.

ÖRTÜŞMEYEN GÖRÜNTÜ PARÇALARININ MOZAİKLENMESİ İÇİN DERİN ÖĞRENME TABANLI HOMOGRAPHY PARAMETRELERİNİN TAHMİNİ

ÖZ

Görüntü mozaikleme, yüksek çözünürlüklü yeni ve büyük bir görüntü oluşturmak için daha geniş görüş alanına (FOV) sahip iki görüntünün birleştirilmesi işlemidir. Mozaikleme için genellikle iki tür yaklaşım kullanılmıştır. Birinci tür yaklaşımlara öznitelik tespitine dayanan geleneksel yöntemler adı verilmektedir. Mozaik kalitesi özniteliklerin sayısına bağlıdır. Geleneksel yöntemler, az sayıda özniteliğe sahip görüntülerde iyi performans göstermemektedir. Denetimli öğrenmeye dayalı görüntü mozaikleme çözümleri güvenilmezdir çünkü bu yöntemler sıklıkla eksik veri sorunuyla karşılaşır. Literatürde bu alanda yapılan çalışmalar genel olarak ortak bir örtüşme alanına sahip olan görüntüleri birleştirmeyi amaçlamaktadır. Bu makalede, ortak örtüşme alanına sahip olmayan görüntüleri kenarların öteleme bilgilerini kullanarak birleştirmeye çalışıyoruz. Bunun için öğrenmeye dayalı bir yöntem öneriyoruz ve derin bir homografi tahmin ağı kullanıyoruz. Ağ, hedef görüntünün referans görüntüye göre yatay ve dikey eksendeki yer değiştirme miktarını tahmin etmek için özel olarak oluşturulmuş bir veri kümesi kullanılarak eğitilir. Deneysel testler, manzara görüntüleri ve bunların Gauss gürültüsü eklenmiş versiyonlarını içeren özel olarak oluşturulmuş bir veri seti kullanılarak gerçekleştirilir. Test sonuçları, ağın iki görüntü arasındaki yatay ve dikey eksendeki yer değiştirme miktarını başarılı bir şekilde tahmin edebildiğini ve dolayısıyla iki görüntünün doğru noktadan birleştirilebildiğini göstermektedir.

Anahtar kelimeler: Bilgisayarlı görme, Derin Görüntü Mozaikleme, Derin Homografi Tahmini, Görüntü Kaydı, Görüntü Harmanlama.

CONTENTS

NOMENCLATURE

Greek Letter Symbols

δ	Change Amount
λ	Amount of Decay
σ	Standart Deviation
-	Sum Operation

Subscripts

max	Maximum
min	Minimum

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Image mosaicing is the process of combining multiple images to create a high-resolution, panoramic image with a wider FOV [21]. Examples of areas where image mosaicing is used include satellite imaging, medical imaging, virtual imaging and games, mobile applications, underwater surveillance etc. The image mosaicing process consists of feature matching, registration and seam removal stages. A well mosaiced image should be high resolution and smooth-edged. Image mosaicing methods in the literature can be grouped into two; the traditional methods and learning-based methods.

1.1 IMAGE MOSAICING WITH TRADITIONAL METHODS

Traditional methods aim to combine images with overlapping area. These methods are based on computer vision. Traditional methods consist of image registration and image blending stages. Registration expresses the geometric transformation between image pairs by feature matching. In the registration stage, geometric transformation estimation is made and images are aligned according to the reference image. The most general case of the transformation is the so called homography. Homography is a matrix with 8 unknowns that contains information about rotation, translation and scaling between a pair of images. In the image registration phase, firstly, feature extraction is performed for two images. Then, the common features in the overlap area of these images are matched. After the feature matching process, the transformation matrix is extracted between the images. Some errors may occur after combining images according to the transformation matrix. Misalignment and ghosting are examples of these errors. To eliminate these errors, blending is performed. Geometric transformation is a method often used when aligning images with a certain common overlap area.

1.1.1 Traditional Image Mosaicing Studies in the Literature

Image Mosaicing is the process of stitching image pairs that have an overlapping area on the 3D scene plane [1]. Mosaicing can be applied in scenes where images are associated with planar homography [22]. If the images show no parallax effect, i.e., the scene is planar, or the camera rotates only around its optical axis [23].

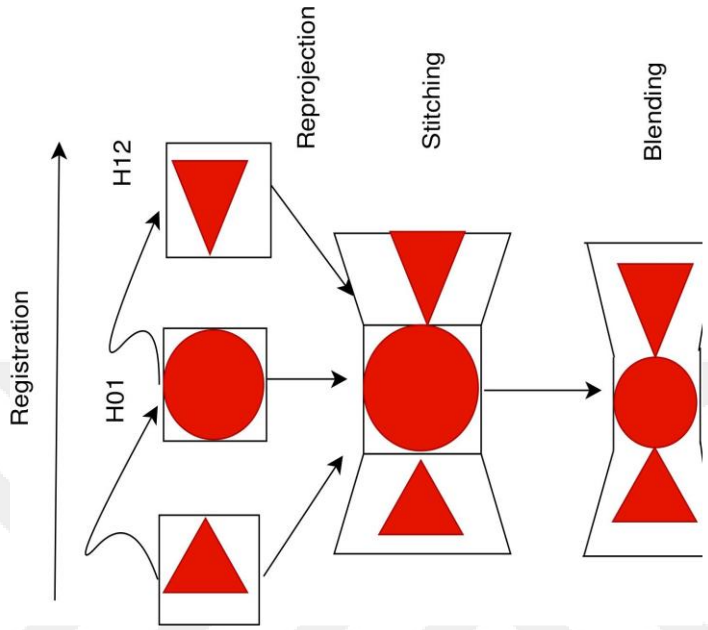


Figure 1.1 Traditional image mosaicing steps. H refers to the homography matrix between image pairs [1]

As seen in Fig.1.1, image mosaicing in traditional methods is done with different image processing algorithms, respectively: registration, reprojection, stitching, and blending [2]. Image registration and blending are the two important steps that affect performance in the mosaicing process dramatically.

In the registration stage, a geometric transformation is estimated between pairs of images in the same plane. Then according to the geometric transformation, the target image and reference image are aligned to each other. As shown in Fig.1.2, registration based mosaicing algorithms in the literature include two different categories: spatial domain and frequency domain-based [2]. Spatial domain-based image mosaicing is divided into two: area and feature-based mosaicing.

Area-based image mosaicing methods can be divided into two groups: Normalized Cross Correlation (NCC) and Mutual Information (MI) based mosaicing. NCC based method calculates similarity between the "windows" in the two images for each displacements defined as [24]. Nasibov et al. [25] developed a mosaicing algorithm that is more sensitive to illumination differences with a brightness correction matrix.

Berberidis et al [26] proposed the spatial cross correlation algorithm. Thus, the displacement between image pairs were calculated and a faster algorithm was presented. In [27], Luna et al. compute similarity between image pairs with MI. In addition, they presented a faster algorithm using a stochastic gradient optimization. To increase the computational efficiency, Dame et.al. [28] proposed a fast algorithm using a B-spline function for normalized mutual probability density.

Frequency-domain based image mosaicing algorithms try to find the optimal transformation parameters between a pair of images.

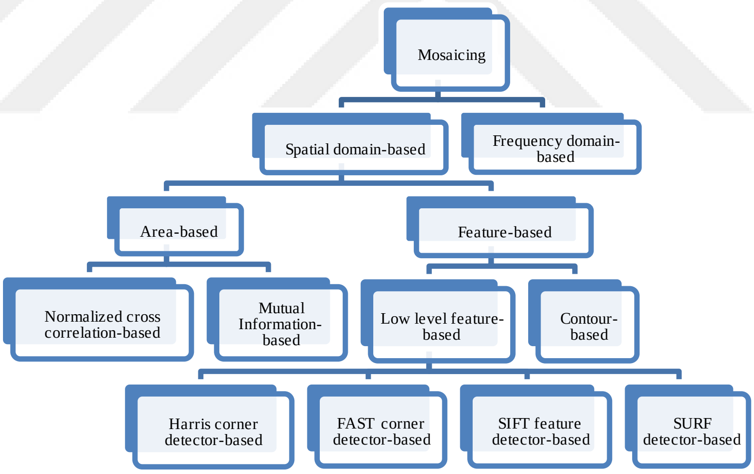


Figure 1.2 Classification based on image registration methods [2]

Image blending algorithms are used in mosicing algorithms to increase efficiency and make a good visual combination. With blending algorithms, the brightness and color transitions in the images are optimized. Thus, problems such as misalignment, ghosting, and camera exposure differences that may arise in mosaic images can be eliminated.

Image mosaicing algorithms can be grouped as shown in Fig.1.3, based on the image blending methods as transition smoothening and optimal-seam based [2]).

Transition-smoothening based mosaicing algorithms are applied to overlapping regions of images and provide seamless transition. These algorithms are divided into 3: feathering, pyramid and gradient-based blending algorithms. Feather-based algorithms perform blending by averaging the pixel values in the common area of image pairs [29]. As an example, the study of Levin et al [30]. can be given. Pyramid-based algorithms provide a more efficient method by converting input images into bandpass pyramids [31]. Xiong et al proposed a pyramid-based method [32]. Gradient-based algorithms provide a method of mosaicing by blending gradients of images. An example of gradient-based algorithms is the study of Szeliski et al [33].

Optimum seam-based mosaicing algorithms try to provide a seamless transition by searching for the most appropriate seam at the points where the images stitch. Examples of optimal seam-based algorithms include a modified difference region method in [34], graph cutting optimization and an algorithm based on watershed segmentation in [35], and Dynamic programming and gray relational analysis-based methods in [36].

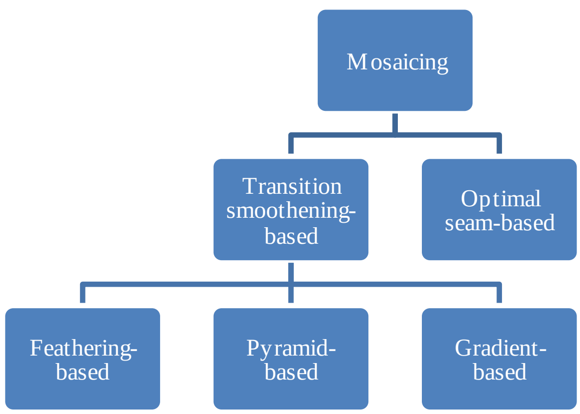


Figure 1.3 Classification based on image blending methods [2]

1.2 IMAGE MOSAICING WITH LEARNING BASED METHODS

Convolutional Neural Network (CNN), a type of regular feedforward neural network which self-learns feature engineering using filters(kernel) optimization. CNN (Convolutional Neural Network) based methods have become widespread recently thanks to their feature extraction capabilities. These methods are used in areas such as object tracking, depth estimation, semantic segmentation, optical flow, voice recognition algorithms etc.

CNN has also started to be used in the field of image mosaicing. Additionally, homography estimation can be made with Deep Convolutional Neural Network method. As shown in Fig.1.4, HomographyNet is a deep neural network that can estimate the homography matrix between two images.

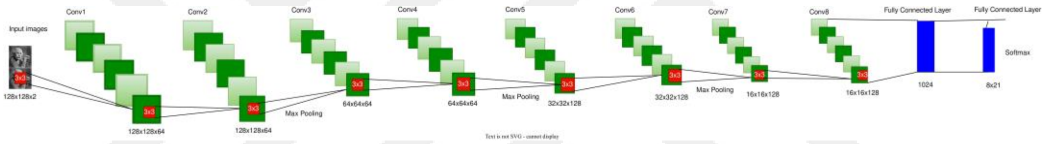


Figure 1.4 Deep Image Homography Estimation (HomographyNet) [3].

1.2.1 Learning Based Image Mosaicing Studies in the Literature

Learning-based image mosaicing methods are divided into view-fixed and view-free methods [37]. View-fixed methods are used in specific areas such as autonomous driving [15], [38], surveillance videos [39]. While end-to-end networks are proposed in these studies, images are not stitched from arbitrary viewpoints. In view-free methods, images can be stitched from arbitrary viewpoints using CNNs. In some studies, CNN's have also been used for feature detection [40], [16]. According to the first complete learning-based method [41], images with arbitrary viewpoints can be stitched in three stages: homography estimation, spatial transformation and content refinement. In addition to this method, the resolution limitation is removed with an edge-preserving deep image stitching method and significant improvement in performance achieved in real scenes [42]. Since we use the homography method in this study, we will also describe the studies on deep homography. In the first proposed deep homography method [3], a homography matrix was estimated using a VGG style network [43] corresponding to eight displacement of the four vertices of the image. Nguyen et al. presented the unsupervised deep homography method [44] using effective unsupervised loss in addition to this architecture [3]. Zhang et al. [45] proposes a a unsupervised network that content-aware, contributing to SOTA performance in small-baseline deep homography. In another study [46], using image pyramids, multi-scale features were detected and homography estimation was performed.

1.3 MOTIVATION OF THIS THESIS

Studies in the field of image mosaicing are generally stitch pairs of images with a common overlap area. In this study, we describe our stitching method for pairs of images that do not have a common overlap area.

We used homography as a basis for this mosaicing process and use a deep learning-based method. When creating a dataset for pairs of images that do not have overlapping areas, we cut a pair of images on a large image so that one image ends where the other image begins. We used only translation while using the homography method in these image pairs that do not include rotation and scaling between them. We trained a deep homography network according to the amount of translation in the horizontal and vertical axes between the image pairs, then stitched the image pairs that have non-overlap area in the test process.

In summary, our method; allows pairs of images that have non-overlapping area to be stitched with horizontal and vertical translation information (without rotation and scaling) using a homography matrix. We perform this mosaicing process with a deep learning-based method.

1.4 ORGANIZATION OF THIS THESIS

In the introduction part of the thesis, a definition of the problem is made and information is given about the previous studies in this field in the literature. In the second stage, detailed information is given about the homography matrix that will be used to solve the problem. In addition, the use of the homography matrix in learning-based studies is also explained. In the third stage, the method used to solve the problem is mentioned. In the fourth stage, different measurements made will be mentioned. In the fifth stage, the results of these measurements will be examined and concluded.

CHAPTER 2

IMAGE MOSAICING TECHNIQUES

2.1 REGISTRATION BASED IMAGE MOSAICING TECHNIQUES

Image registration allows establishing a geometric relationship between a pair of images in the same scene. In order to perform image registration, a geometric transformation that allows aligning the target image with the reference image must be obtained. Image registration forms the basis of the mosaicing process. It calculates an optimum geometric transformation between pairs of images taken from different times, perspectives and sensors and ensures their alignment to a common plane.By matching the features extracted from the images as in Fig.2.1, the phase correlation feature is used to extract the frequency relationship.Registration based image mosaicing techniques are divided two as spatial domain and frequency domain, as in Fig.1.2.

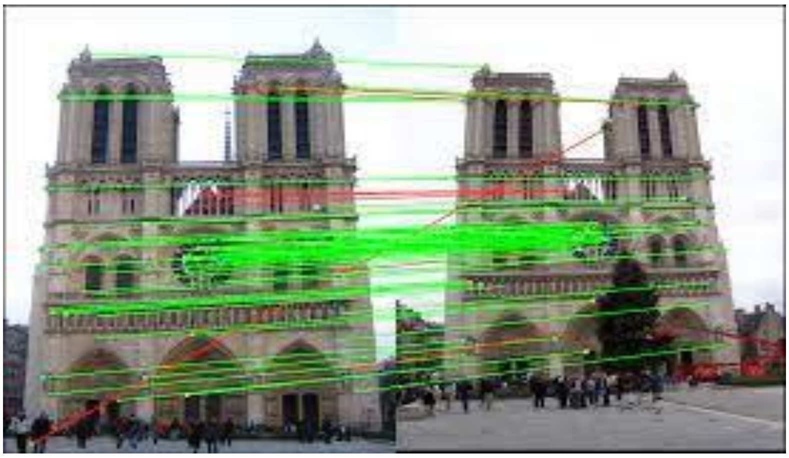


Figure 2.1 Example of Feature Extraction and Matching [4].

2.1.1 Spatial Domain Image Mosaicing

Spatial area-based mosaicing algorithms are divided into area-based or feature-based.

2.1.1.1 Area Based Image Mosaicing

Area-based mosaicing algorithms are called pixel-based mosaicing because they match from pixels instead of matching features as in Fig.2.2. Area-based image mosaicing algorithms can be divided two as normalized cross-correlation and mutual information-based mosaicing.

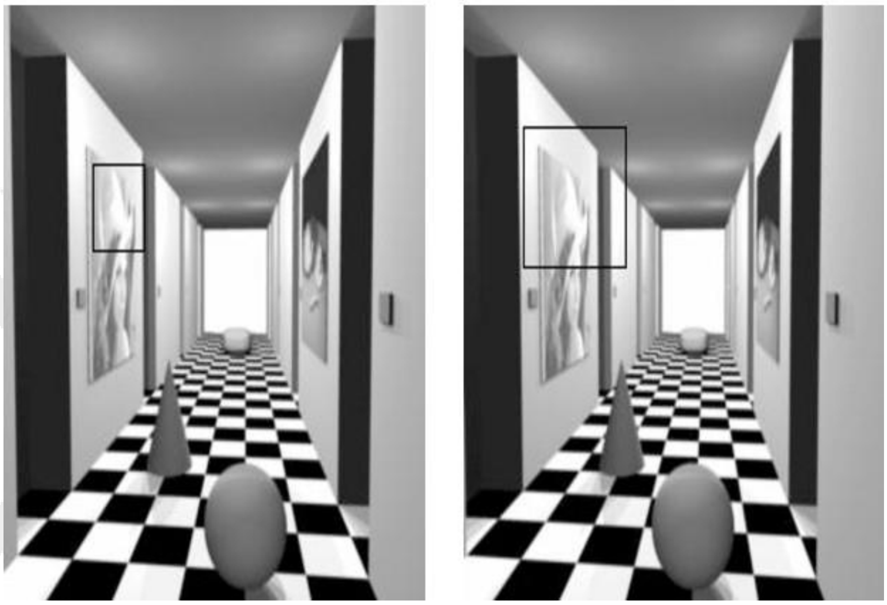


Figure 2.2 Example of Area Based Matching [5]

Normalized Cross Correlation-based method select a window on the target and reference images and calculate the similarity between these windows.The shift parameter represents the geometric transformation between two images. These methods are computationally easy but slow.

$$\mathrm{NCC(u)} = \frac{\Sigma \mathrm{i}[\mathrm{Ir(xi} - \mathrm{I}_n{}')][\mathrm{It(xi+u)} - \mathrm{I}_t{}']}{\Sigma \mathrm{i}[\mathrm{Ir(xi} - \mathrm{I1}\,)^{t}][\mathrm{It(xi+u)} - \mathrm{I}_t{}^{2}]} \quad \textbf{(2.1)}$$

where

$$\mathrm{I}' = \frac{1}{N}\sum I_r\,(xi) \quad \textbf{(2.2)}$$

$$\mathrm{I}' = \frac{1}{N}\sum I_t\,(xi+u) \quad \textbf{(2.3)}$$

Unlike NCC, Mutual Information-based methods measure the similarity between two images using the brightness information of the images.

$$M(I_r, I_t) = E(I_r) + E(I_t) - E(I_r, I_t) \quad \textbf{(2.4)}$$

$E(I_r)$:Entropy of the reference image.

$E(I_t)$:Entropy of the target image.

$E(I_r, I_t)$:Joint entropy between reference and target images.

$$E(I_r) = -\Sigma P_{I_r}(g)\log(P_{I_r}(g)) \quad \textbf{(2.5)}$$

where g are the possible gray level values of first image according to PDF(probability distribution function).

$$E(I_r, I_t) = -\Sigma P_{I_r I_t}(g, h)\log(P_{I_r I_t}(g, h)) \quad \textbf{(2.6)}$$

where h indicates possible gray values of second image and $P(I_r, I_t)$ represent joint PDF according to g and h.In order to have a good alignment between the two images, the MI value must be high.

MI-based mosaicing methods are less sensitive to brightness changes between images. Like NCC-based methods, MI-based methods are computationally slow and require a high amount of overlap between images.

The Fig.2.3 shows the alignment of an image pair according to the NCC and MI based mosaicing methods.

Figure 2.3 Comparison of NCC and MI based mosaicing. [6]

2.1.1.2 Feature Based Image Mosaicing

These methods do not require a large amount of overlap area. These algorithms use low-level features such as pixels, colors, edges, corners, hisograms that are effective and common throughout the image.Harris Corner Detector, FAST Corner Detector, SIRF Feature Detector and SURF Feature Detector can be given as examples of low level feature extraction methods.

Harris Corner Detector: Harris corner detector finds corner points in images. Its formulation is expressed below [47].

$$E(u,v) = \Sigma w(x,y)[I(x+u,y+v) - I(x,y)]^2 \qquad \textbf{(2.7)}$$

where $w(x,y)$: Function of window, $I(x+u, y+v)$: Shifted intensity, $I(x, y)$: Intensity. The window function can be a Gaussian window or rectangular.For corner detection, the E(u,v) function must be maximized, for this the image derivatives Ix and Iy calculated by the Sobel operator in the horizontal(x) and vertical(y) directions are used.

$$E(u,v) = [uv]M \begin{bmatrix} u \\ v \end{bmatrix} \qquad \textbf{(2.8)}$$

$$M = \Sigma w(x,\ y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \qquad \textbf{(2.9)}$$

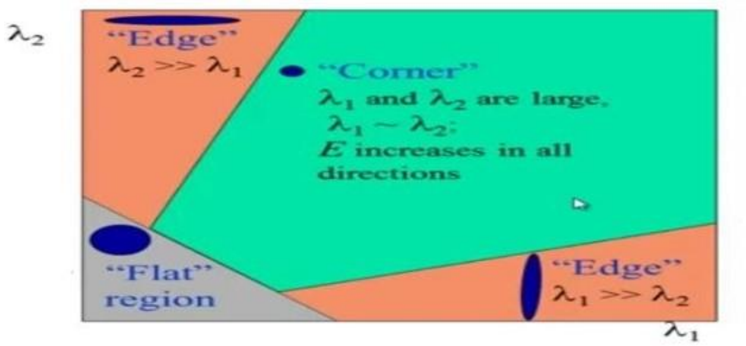


Figure 2.4 Harris representation [7]

$$R = \det(M) - k(\mathrm{trace}(M))^2 \tag{2.10}$$

det(M)= $\lambda_1\lambda_2$, trace(M)=$\lambda_1 + \lambda_2$, λ_1 and λ_2 is eigenvalues of M.

When λ_1,λ_2 and R are small, the region is flat. When λ_1»When λ_2 and vice versa, R is<0, the region is edge. When λ_1,λ_2 and R are large, the region is a corner. The figure associated with the Harris corner detector is given below.

An example visual figure for corner detection with the Harris corner detector is also given in Fig.2.5.

FAST Corner Detector: Since the FAST algorithm is computationally faster and more efficient than other low-level feature detection algorithms, it is more suitable for real-time algorithms. The FAST algorithm can be summarized as follows. 1)A pixel is selected on the image and its intensity is called I_p.

2) An appropriate threshold value (T) is determined.

3) A circle of 16 pixels is created around the tested pixel.

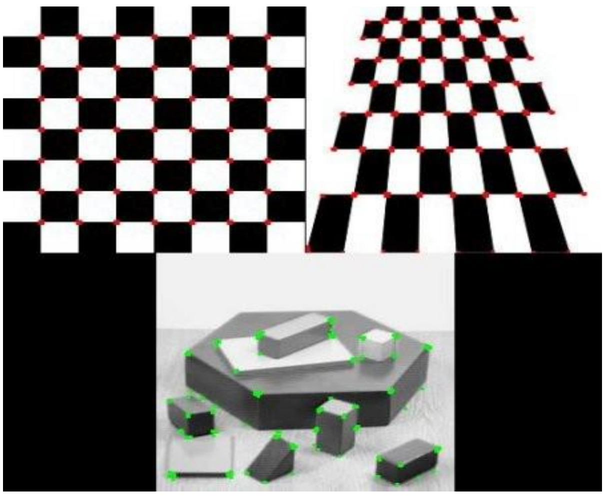


Figure 2.5 Example of Harris Corner Detector [7]

4) In this circle, pixels with brightness values brighter than $I_p + T$. and darker than $I_p - T$. are called corners. An example visual figure for corner detection with the FAST algorithm is also given in Fig.2.6.

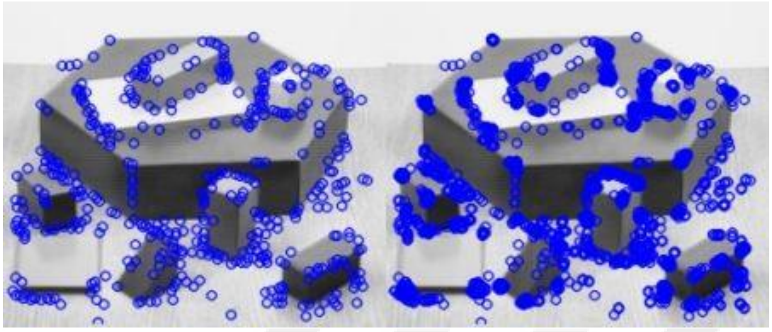


Figure 2.6 Example of FAST Corner Detector [8]

SIFT Feature Detector: The SIFT algorithm detects key points on the image. SIFT descriptor is not affected by translation, rotation, scaling changes and is resistant to brightness changes. The SIFT algorithm can be summarized as follows.

1) Firstly, for the image; scale space is created using the Gaussian filter as follows.

$(S(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ G=Gaussian filter, I=input image, *=convolution operator

2) The difference of Gaussian(DoG) images is calculated at each octave.

$(DoG(x, y, \sigma) = S(x, y, k\sigma) - S(x, y, \sigma)$

3) In the next step, low-contrast keypoints and edges are eliminated using accurate keypoint localization.Gradient direction formula as follows.

$(\theta(x, y) = \arctan(((S(x, y+1) - S(x, y-1))/((S(x+1, y) - S(x-1, y)))$

4) Key points are created based on the orientation histograms of neighboring pixels.

5) 128-dimensional vector which normalized descriptor is calculated for each key point.

Outliers are eliminated with the RANSAC algorithm and images are combined using transformation parameters. SIFT feature detector are available for high resolution images but are costly.An example visual figure for corner detection with the SIFT algorithm is also given in Fig.2.7.

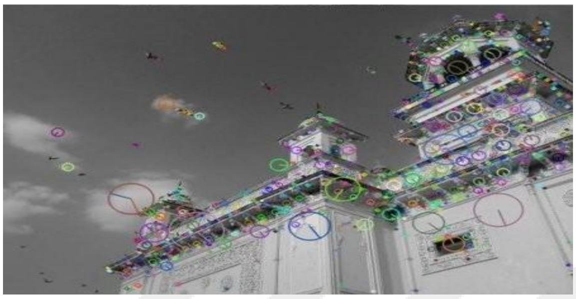


Figure 2.7 Example of SIFT Feature Detector [9]

SURF Feature Detector: Like SIFT, the SURF algorithm uses scale space theory and estimates local maxima in different scale spaces by calculating the Hessian matrix of the integral image.The general formula of the Hessian matrix is as follows.

$$H(x,y,\sigma)=\begin{bmatrix} Sxx(x,y,\sigma) & Sxy(x,y,\sigma) \\ Sxy(x,y,\sigma) & Syy(x,y,\sigma) \end{bmatrix} \quad \textbf{(2.11)}$$

While finding the Hessian matrix, Gaussian filtering is performed using the S_{xx}, S_{xy}, S_{yy} are convolutions of image point (x,y) with Gaussian second order following box filters in Fig.2.8.

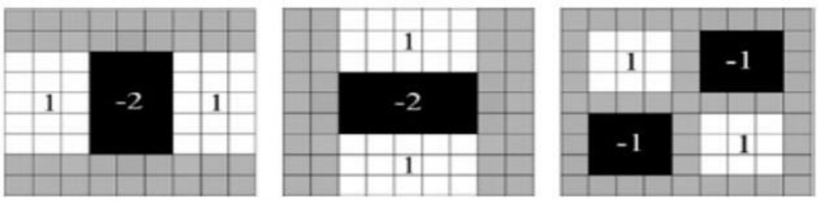


Figure 2.8 Left to right x,y and xy direction box filter [10]

After calculating the local maximum point with the Hessian matrix, a direction is assigned to each key point and a vector is created and the longest vector represents the orientation of the key point. Small subregions are then created and a 64-dimensional descriptive vector is created for each subregion. Outliers are eliminated with the RANSAC algorithm. The SUFT algorithm is faster than SIFT, but performs poorly in special cases such as color, illumination and affine transformation. The Fig.2.9 shows an example of feature detection with the SURF algorithm.

Figure 2.9 An example of SURF algorithm [10]

2.1.1.3 Contour Based Image Mosaicing

Such mosaicing algorithms detect more natural features for human perception and often extract shapes from the image and determine their size, direction and location. Then, they find relationships by matching. After incorrect matches are eliminated, the transormation parameters are calculated and the blending process is performed. In high level mosaicing algorithms, the computational cost is higher and they show higher performance in complex movements. The Fig.2.10 shows an example of contour detection algorithm.

Figure 2.10 An example of contour detection

2.1.2 Frequency Domain Image Mosaicing

Unlike spatial domain-based image mosaicing algorithms, it estimates the transformation parameters between images by calculating in the frequency domain using phase correlation.

$$I_t(x, y) = I_r(x - x_0, y - y_0) \quad \textbf{(2.12)}$$

where $I_r(x, y)$ and $I_t(x, y)$ are reference and target images respectively and x_0 and y_0 are translational values. The Fourier transform for the image pair can be formulated as follows.

$$F_t(u,v) = F_r(u,v).e^{(-j(wx0+vy0)} \quad \textbf{(2.13)}$$

The cross power spectrum for image pairs $e^{(-j(wx0+vy0))}$

$$\frac{Fr*(u,v)Ft(u,v)}{|Fr*(u,v)Ft(u,v)|} \quad \textbf{(2.14)}$$

where F_r* is complex conjugate of $F_r(u, v)$. The shift theorem calculates displacement using the phase difference between images. Using these displacement parameters, the image pair is stitched and a mosaic is obtained.Frequency-based mosaicing algorithms are efficient due to the use of Fast Fourier Transform(FFT), but are sensitive to noise and require a large amount of overlap area.Fig.2.11shows an example of the conversion of an image from the spatial plane to the frequency plane.

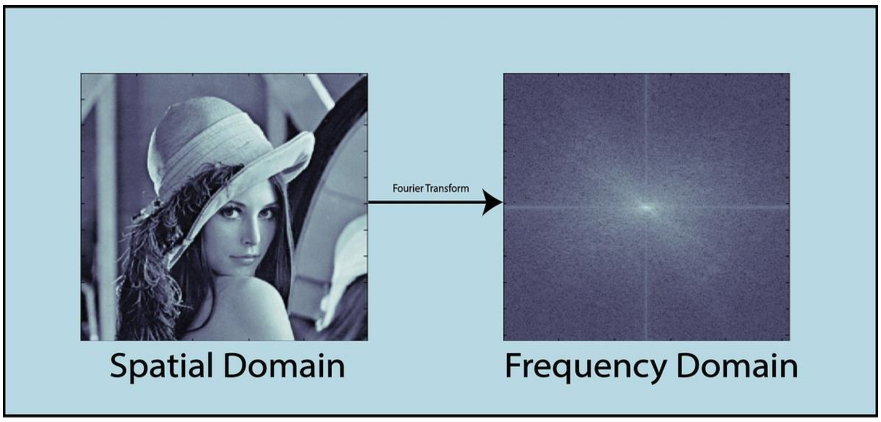


Figure 2.11 Spatial domain to frequency domain [12].

2.2 BLENDING BASED IMAGE MOSAICING TECHNIQUES

The purpose of image mosaicing is to combine pixels in overlapping regions and preserve pixels in non-overlapping regions. However, as a result of geometric and photometric misalignments, seams may appear in the border regions between images.In the image mosaicing process, blending algorithms are required to prevent inconsistencies that may occur due to the presence of moving objects, geometric misalignments, and lighting differences. In this way, it is aimed to minimize the color and difference of brightness between the image pairs.In Fig.2.12, you can see an example of the artifact created as a result of the mosaicing process.Some blending algorithms are required to eliminate such errors.

Figure 2.12 An example of artifact [13]

2.2.1 Transition Smoothing-based Image Mosaicing

This type of algorithms tries to reduce the seam visibility by softening the overlapping regions of the merged images, thus making the transition between images indistinguishable. Popular methods that use transition-smoothing for blending operations include feathering, pyramid gradient-based blending.

2.2.1.1 Feathering-Based Image Mosaicing

This type of algorithms takes an average value in the overlapping pixels of the images and performs the blending process. However, this process may show incorrect results in images with moving objects. Therefore it is based on distance. If pixels near the center of the image, they weighted heavily. If pixels near the edge of the image they weighted lightly. Its formulation is as follows.

$$w_k(x) = ||\text{argmin}||yy||I_k{}' (x + y)\text{isinvalid}.|| \quad \textbf{(2.15)}$$

where $I_k{}'$ is warped image and $w_k(x)$ is weight of the images. It is difficult to smooth out the exposure difference and achieve a balanced transition at low frequencies. Therefore, this method causes artifacts at low frequencies.The Fig2.13 shows the difference in mosaicing between feathering-based blending and simple averaging-based blending algorithms.

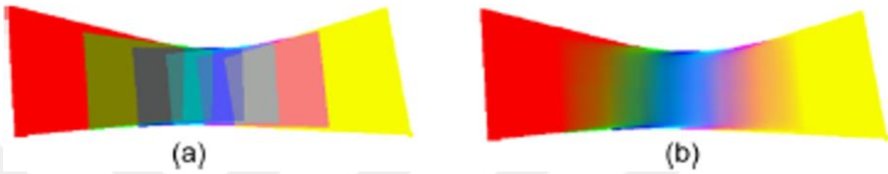


Figure 2.13 a) Simple averaging blending, b)Feathering blending. [2]

2.2.1.2 Pyramid-Based Image Mosaicing

Pyramid-based mosaicing algorithms convert input images into bandpass pyramids and a mask image is created. The mask image is then converted into a low pass pyramid using a Gaussian kernel. Its general formula is as follows.

$$LO(x,y) = GM(x,y) * LI_1(x,y) + (1 - GM(x,y)) * LI_2(x,y) \quad \textbf{(2.16)}$$

where $LI_1(x, y) and LI_2(x, y)$ are Laplacian pyramids of images. $GM(x, y)$ is Gaussian pyramid of mask image and $LO(x, y)$ is Laplacian pyramid of the output image.Fig.2.14 shows the sampling of an image with a low pass pyramid and a band pass pyramid.

Figure 2.14 a)Low-pass pyramid, b)Band-pass pyramid. [2]

2.2.1.3 Gradient-Based Image Mosaicing

In gradient-based methods, the transitions between seams are set to zero and the gradients of the images are blended. Thus, color differences are softened.

2.2.2 Optimal Seam-Based Image Mosaicing

Optimal seam-based mosaicing algorithms aim to minimize photometric differences by calculating the most optimal position for the stitch line. In this way, a smooth transition between images is ensured.Optimal seam-based mosaicing algorithms also take into account objects and parallax in the scene. Thus, it can distinguish exposure differences.Fig.2.15 shows an example of a seamless mosaicing of three images.

Figure 2.15 Example of image mosaicing with no seam [14].

2.3 LEARNING BASED IMAGE MOSAICING TECHNIQUES

Computer vision-based methods can be easily affected by many parameters such as lighting, scene structure change, and can give erroneous results in many cases, including image pairs with low overlap rate. For this reason, learning-based methods are becoming common in the field of image processing. Creating a real data set in image mosaicing is a difficult process. In addition, combining images with deep learning-based methods in image pairs with scenes with large parallax and low overlap ratio is a difficult process. For this reason, the development phase for deep learning-based image mosaicing methods continues today.Learning-based image mosaicing methods are divided into view-fixed and view-free methods [37].

2.3.1 View-fixed Mosaicing Methods

View-fixed image mosaicing methods are widely used in areas such as autonomous driving and surveillance videos. In this method, images are stitched from fixed views using end-to-end networks. Viex-fixed image stitching methods work on full-scale images. In these methods, a convolutional feature extraction network is designed by generating locations and descriptors for feature points. As shown in Fig.2.16, the model has a shared encoder that enables feature extraction, and the size of the image is reduced thanks to the max pooling layers found here. There are also 2 decoders in the network. One of these decoders is called Feature Point Detector and allows learning feature detectors on the image. The other one is called descriptor decoder and enables the learning of feature point descriptors. The input and output of the network have the same size to ensure location compatibility.

2.3.1.1 Shared Encoder

In the encoder stage, 8 convolution layers are used to extract features and 3 max-pooling layers to reduce image resolution. While the first 4 convolution layers have 64-dimensional channels, the last 4 convolution layers have 128-dimensional channels. After every 2 convolution layers there is a maximum pooling layer of size 2x2. All convolutional layers use ReLu as activation function. Each encoder affects an area of size 8x8.

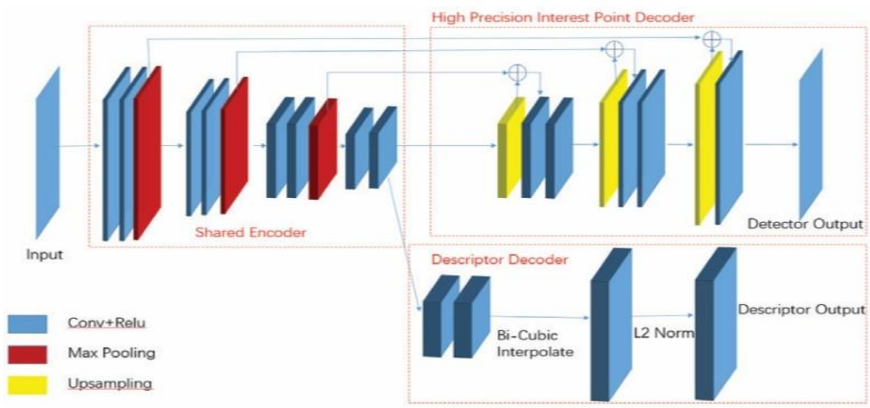


Figure 2.16 Example of view-fixed network [15].

2.3.1.2 Descriptor Decoder

Similar to the SIFT algorithm, it creates a 256-dimensional descriptor vector to extract features from an image of size HxW. The features are filtered with a convolutional kernel of size 3x3x128 to create the output. With the bicubic interpolation method, the image size is preserved. Then, a descriptive vector is created for each pixel.

Figure 2.17 Examples of image stitching results with view-fixed learning methods.

2.3.1.3 Feature Point Decoder

It is very important to accurately estimate the positions of feature points in the image stitching process. To preserve the location information generated by the pooling layer, a method that takes into account low-level visual features in addition to high-level semantic features is adopted. Low-level feature maps are concatenated into features by bilinear interpolation. The position of each feature is preserved by using the sigmoid as the activation function. Thus, in addition to high-level features on the image, low-level visual feature maps are created.Fig.2.17 gives an example of a stitching process with the view-fixed image stitching method.

2.3.2 View-free Mosaicing Methods

In view-free methods, images can be stitched from arbitrary viewpoints using CNNs. In some studies, CNN's have also been used for feature detection [40], [16]. According to the first complete learning-based method [41], images with arbitrary viewpoints can be stitched in three stages: homography estimation, spatial transformation and content refinement. In addition to this method, the resolution limitation is removed with an edge-preserving deep image stitching method and significant improvement in performance achieved in real scenes [42]. In view-free methods, the semantic features

of each pixel are calculated by the neural network and the semantic contribution values of each pixel are ranked from high to low and an image mosaic is created. Fig.2.18 shows the image of the CNN network used to extract features in view-free methods.

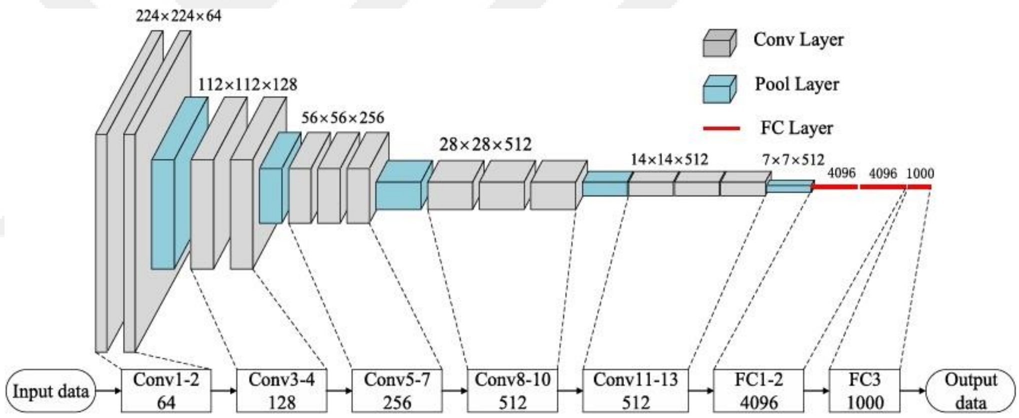


Figure 2.18 Example of view-free network [16]

After every 2 convolutional layers, 1 pooling layer is used and at the end, 2 fully connected layers are used.Fig.2.19 gives an example of image mosaicing using the view-free method. The feature extraction process, based on the convolution neural network method, consists of 5 steps.These are training network for feature detecting, extracting semantic feature points by network,creating matching point by matching feature vectors and remove incorrect matches that do not comply with the scale, rotation and slope features,obtaining a mosaic image by interpolation and affine transformation method respectively.Thus, high-level semantic features are extracted by CNN.

Figure 2.19 Example of view-free image stitching result [16].

CHAPTER 3

DEEP HOMOGRAPHY ESTIMATION

3.1 INTRODUCTION

In this study, we are mosaicing images that have a non-overlap area with a learning-based method. We provide this mosaicing process with the homography method. We train our network according to the homography matrix between pairs of images in our dataset. The non-overlap image mosaicing process explains of 3 steps homography estimation, network architecture, loss function and dataset preparation respectively.

3.2 HOMOGRAPHY ESTIMATION

Homography is a matrix that aims to find the geometric relationship between two images taken from different perspectives in the same scene in the fields of computer vision and image processing. Homography estimation aims to align and stitch images. For homography estimation, a sequence of points is found between two images and a mathematical model is created accordingly. Homography estimation can be done with RANSAC and various machine learning based methods. Fig.3.1 explains the homography relationship between two images.

Point x in the reference image corresponds to point x' in the target image. This point also has a location in the common plane of the two images. The relationship between points x and x' can be expressed by the homography matrix. According to this; Rotation, translation and scale information between two images are expressed by the homography matrix.

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.1}$$

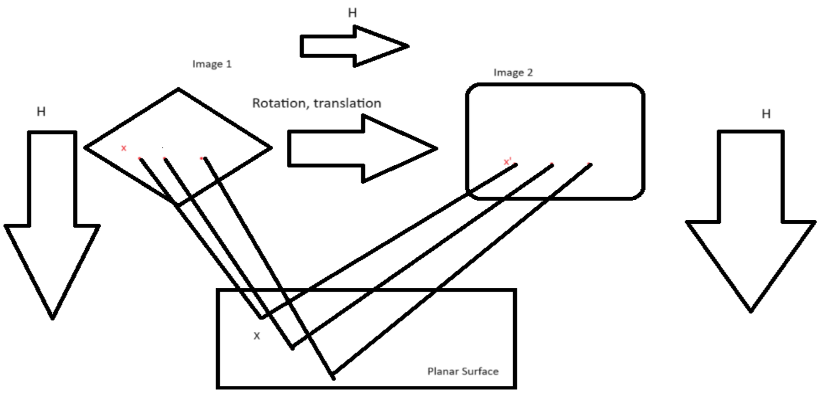


Figure 3.1 Homography relationship between two images.

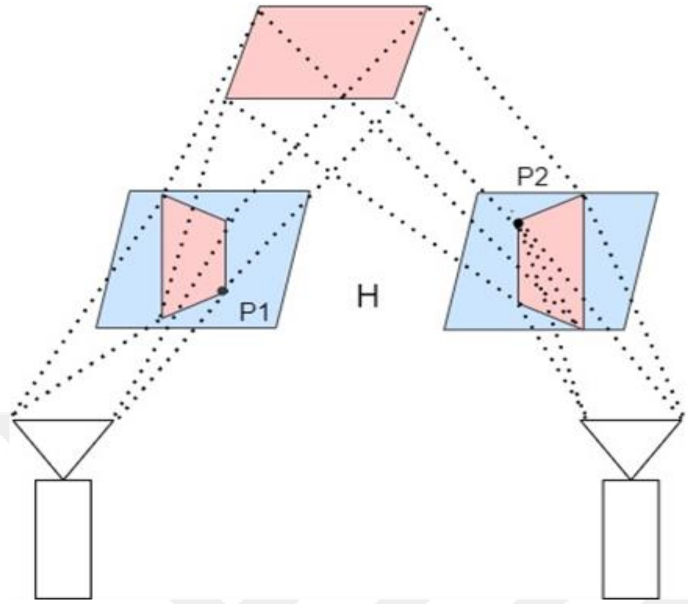
The homography maps [x, y], the pixels in the reference image, to [x', y'], the pixels in the target image, and is defined up to scale (see Eq.**3.1**). The homography matrix is divided into submatrices as rotation and translation. [H_{11} H_{12} ; H_{21} H_{22}] submatrix represent the rotational terms and [H_{13} H_{23}] is the translational offset.

3.2.1 4 Point Parametrization

According to equation **3.1**, the homography matrix contains 9 parameters. Here, [H_{11} H_{12} ; H_{21} H_{22}] contains rotation terms and [H_{13} H_{23}] contains translation terms. This complexity in the homography matrix causes difficulties in optimization problems. For this reason, the 4 point parametrization method is used by . [48]. 4 point parametrization is based on the theory of displacement of 4 corners in an image. In our own study, we perform homography estimation based on the 4-point parametrization method.

As shown in Fig.3.2, 4-point parameterization can be used to express the coordinate difference between the vertices in two images. 4-point parameterization is used in traditional homography estimation methods [48] and is also suitable for deep homography estimation methods. For example, if $\Delta_{u1} = u1' - u1$ be the u-displacement for the corner points, 4-point parameterization is formulated as follows [3]:

$$H_{4point} = \begin{bmatrix} \Delta_{u1} & \Delta_{v1} \\ \Delta_{u2} & \Delta_{v2} \\ \Delta_{u3} & \Delta_{v3} \\ \Delta_{u4} & \Delta_{v4} \end{bmatrix} \quad \textbf{(3.2)}$$



Point	x	y	z
P1	u1	v1	1
P2	u1'	v1'	1

Figure 3.2 4 point parametrization

If the coordinate changes in the horizontal and vertical axes between the image pairs are known, H_{4point} can easily convert to homography matrix. This can be defined as the direct linear transform (DLT) algorithm [49] In our study of non-overlap image mosaicing, there is no rotational and scaling difference between the image pairs. Therefore, we only use the translational terms in the horizontal and vertical axis in training and testing processes. While creating the dataset, firstly p=(x,y) point is selected in a large I image and a 320x240 image is cut from the beginning of this point. Let's call this image patch A. Then, a random amount of translation is generated for the horizontal and vertical axis so that there is no overlap between the images. These translation values are summed with the coordinates of the p point and a new p=(x+tx,y+ty) point is created. A new 320x240 image is cut from the beginning of the p point, which is called patch B. The coordinate difference between the 4 vertices of the A and B images is expressed as H_{AB} with the 4-point parameterization as Fig.3.3.

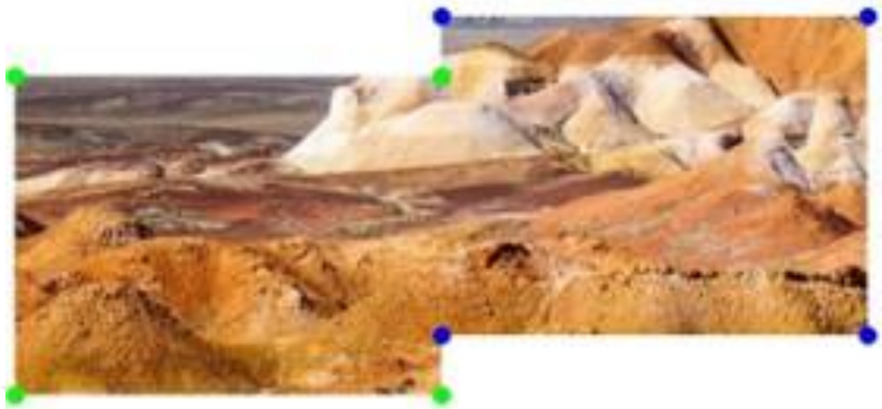


Figure 3.3 Dataset Creating.

According to Fig.3.3, green points represent the corners of the first image and blue points represent the corners of the second image.With the 4-point parametrization method, a homography matrix is created between the two images by calculating the displacement of each corner in the 1st image according to the second image.The homography matrix that we extract between two images is the ground truth homography matrix for us. In the next stage, we will use this ground truth matrix for the loss function that we will calculate difference between the homography matrix produced by our deep network and ground truth homography matrix.

3.3 NETWORK ARCHITECTURE

At this stage, we create a network that generates homography output.Our network is based on the VGGNet [43] in Fig.3.4 and HomographyNet [3] in Fig.3.5 architectures.

3.3.1 VGGNet

VGGNet is a typical deep Convolutional Neural Network (CNN) design with multiple layers. It stands for Visual Geometry Group. According to the number of layers it contains; The model with 16 convolutional layers is called VGG-16, and the model with 19 convolutional layers is called VGG-19. VGGNet is a deep neural network. New generation object recognition models are also created with VGG architecture. VGGNet performs better on various datasets such as ImageNet. [50], which was previously used in the field of object recognition. VGGNet is one of the image processing architectures frequently used today.

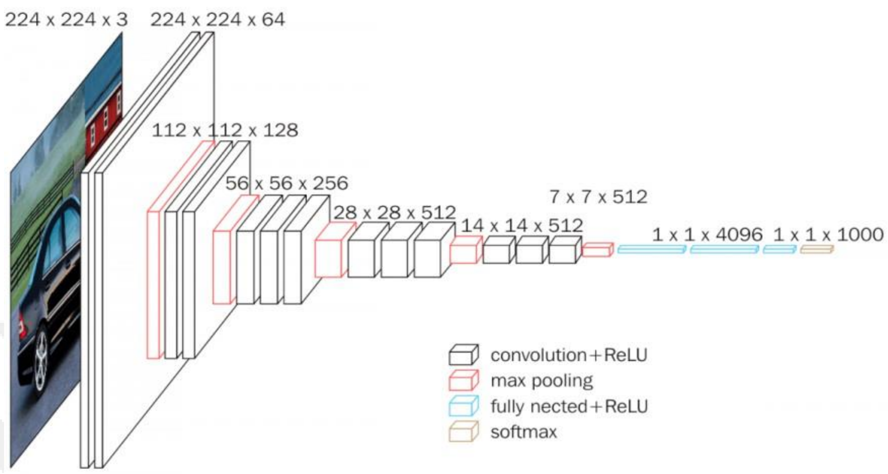


Figure 3.4 VGGNet Architecture [43]

3.3.1.1 VGG-16:

The convolutional neural network model, called VGGNet and containing 16 layers, is referred to as VGG16. It was developed by K. Simonyan and A.Zisserman. In the ImageNet dataset, which contains around 15 million images of approximately 1000 species, the VGG16 model reaches a test accuracy of approximately percantage of 92.2. By using 3x3 kernel size filters instead of large kernel size filters, it outperformed other architectures used in this field, such as AlexNet. [51]. The VGG16 model is trained using Nvidia Titan Black GPU, and as a result of this training, it can distinguish animals such as cats, dogs, mice, as well as objects such as keyboards, balls, and tables.

3.3.1.2 VGG-19:

The only difference between the VGG19 model and the VGG16 model is that VGG19 has 19 layers, other than that it has the same features as the VGG16 model.

There are thirteen convolutional layers and three fully connected layers in the structure of the VGG network. VGGNet architecture consists of the following elements.

Inputs: VGGNet has image inputs of 224x224 pixels. For this purpose, 224x224 images were cut from the images in the reference ImageNet dataset and the VGGNet network was fed with these images.

Convolutional Layers: In VGG convolutional layers, the smallest possible receptive field is used to record left-to-right and top-to-bottom motion. Additionally, the input is linearly transformed with 11 convolution filters. According to AlexNet, the most important factor that shortens the training time of VGG is the ReLu unit. ReLu, which is a linear activation function, is a linear function that gives the output of the input if the input is positive, and if the input is not positive, it gives 0 as the output.

Hidden Layers: ReLu is used in the hidden layers of the VGG network.

Fully Connected Layers; VGGnet has three fully connected layers. The first two of these have 4096 channels, while the third layer has 1000 channels representing each class.

3.3.2 HomographyNet

HomographyNet network that generates homography output. HomographyNet has a structure similar to the VGGNet architecture in Fig.3.4. It uses 3x3 convolutional blocks in addition to BatchNorm and RELU. Two input images feed a network, causing it to produce a homography matrix. In HomographyNet, the input images are grayscale and 128x128x2. HomographyNet is a Deep Convolutional Neural Network that enable homography estimation between image pairs. Network use 8 convolutional layers in the architecture, with 1 max pooling layer (2x2, stride is 2) after 2 convolutional layers. The first 4 of the 8 convolutional layers have 64 filter numbers and the last 4 have 128 filter numbers. After the last 2 convolutional layers, HomograpyhyNet use 2 fully connected layers. After the final convolutional layer and the following first fully connected layer with 1024 units, a dropout with probability 0.5 is applied.The last layer of HomographyNet uses the classification network.

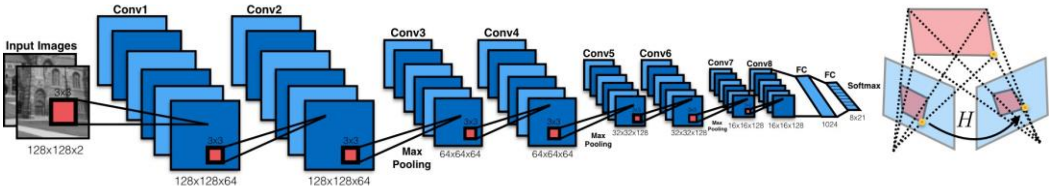


Figure 3.5 HomographyNet Architecture [3].

3.3.3 Our Network

At this stage, we create a network that generates homography output. Our network 4.3 has a structure similar to the HomographyNet architecture in Fig.3.5. Our network has 3x3 convolutional blocks in addition to BatchNorm and RELU as HomographyNet. Two input images feed a network, causing it to produce a homography matrix. In our network, the input images are grayscale and 128x128x2 in size as in HomographyNet. We use a total of 8 convolutional layers in the architecture, with 1 max pooling layer (2x2, stride 2) after every 2 convolutional layers. The first 4 of the 8 convolutional layers have 64 filter numbers and the last 4 have 128 filter numbers. After the last 2 convolutional layers, we use 2 fully connected layers. After the final convolutional layer and the following first fully connected layer with 1024 units, a dropout with probability 0.5 is applied.

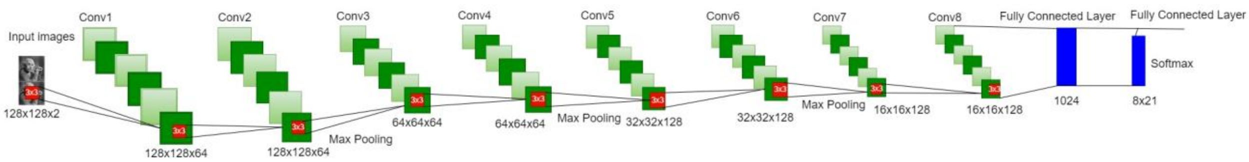


Figure 3.6 Our Network Architecture.

While our network architecture has the same architecture as HomographyNet until the last layer, in the last layer we use a regression network [3] instead of a classification network(Fig. 3.7).

The regression network uses Euclidean loss(L2) and 8 real-valued numbers at the end of the training process. This network, which is simple in formulation, may not always be used because it is not based on a certainty.

The formula of loss function in regression network:

$$\text{Loss : Euclidean(L2)} = 0.5 * |p(x) - q(x)| \quad \textbf{(3.3)}$$

The classification network has a softmax in the last layer and uses a quantization scheme, thus use the cross-entropy loss during training. It is more useful to use on image pairs that contain rotational changes. We do not use the classification network because our study doesn't include rotational changes. De tone et al. in their study; for each of the 8 output dimensions,they chose to use 21 quantization bins which results in a final layer with 168 output neurons.

The formula of loss function in classification network:

$$\text{Loss:Cross-Entropy} = -\textstyle\sum_x p(x) \log q(x) \quad \textbf{(3.4)}$$

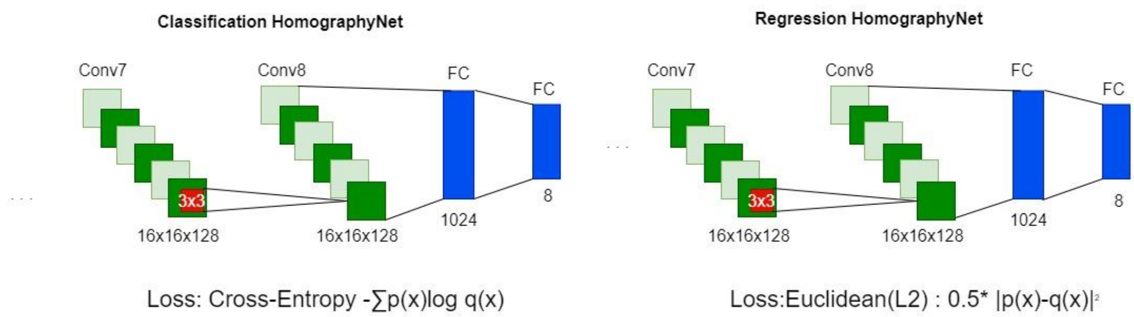


Figure 3.7 Classification Network vs Regression Network [3].

3.4 LOSS FUNCTION

The loss function used in mathematical optimization processes is also called the cost function. It allows expressing the value of one or more variables in an event as a real number representing the cost. In optimization, the loss function is tried to be minimized. The opposite of the loss function is the objective or profit function, and these functions are tried to be maximized simultaneously.In machine learning and deep learning processes, networks are trained according to certain loss functions. The network learns its error according to the loss function at the end of each iteration and tries to minimize this error by reducing it at the end of each iteration. Loss functions used in machine learning are divided into two according to their use in classification and regression networks. Let's give some examples of loss functions used in classification and regression networks.

3.4.1 Classification Based Loss Functions

The two most used loss functions are binary cross-entropy and hinge losses.

3.4.1.1 Binary Cross-Entropy Loss/Log Loss:

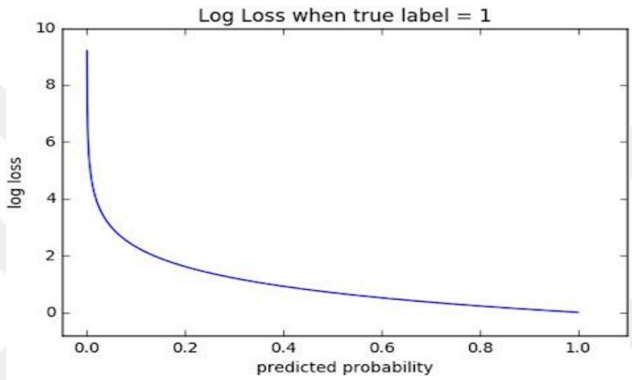


Figure 3.8 Binary Cross-Entropy Loss Function Graph [17]

It is the most commonly used loss function for the classification. The predicted output with a probability value in the range of 0-1 measures the performance of the classification network,and the entropy loss decreases as the predicted probability approaches the true value.

If number of classes is equal to 2, it's binary classification.

$$Loss : LCE = -\frac{1}{M} \ (v_i.log(v_i) + (1 - v_i).log(1 - v_i)) \tag{3.5}$$

If number of classes is greater than 2, it's multi-class classification.

$$Loss : LCE = -\frac{1}{M}(v_i.log(v_i)) \tag{3.6}$$

The graph of the cross entropy function is shown in Fig.3.8.

3.4.1.2 Hinge Loss:

Hinge Loss an alternative to the cross-entropy loss function in classification networks and was developed for the evaluation of support vector machine (SVM) models.It is based on the principle of penalizing incorrect predictions. The graph of the hinge loss function is shown in Fig.3.9.

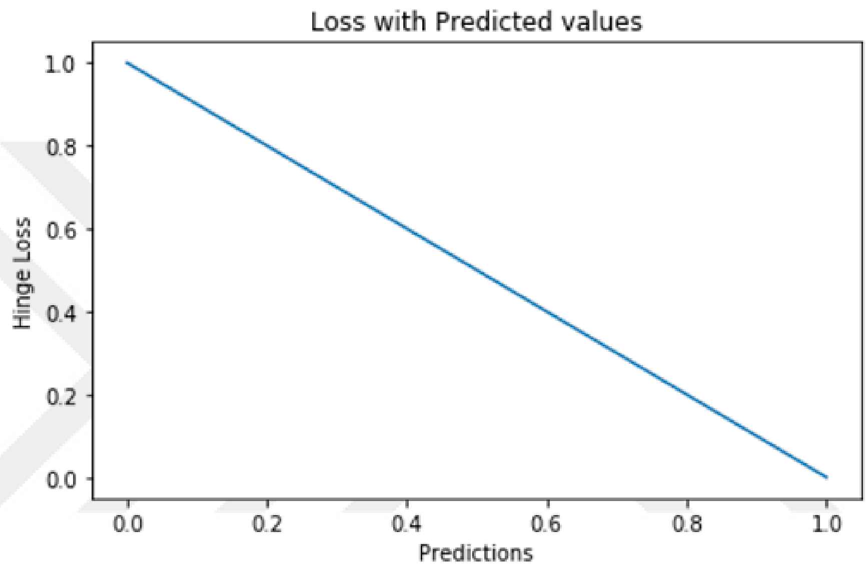


Figure 3.9 Hinge Loss Function Graph [17].

3.4.2 Regression Based Loss Functions

The two most used loss functions are mean-square error and mean absolute error loss.

3.4.2.1 Mean Square Error/Quadratic Loss/L2 Loss:

The Mean Square Error(MSE) loss function is a function that averages the squares of the differences between the ground-truth value and the predicted value. In the MSE loss function, the model is penalized by squaring large errors, and therefore the function is less robust to outliers. Therefore, it is a loss function that should not be used if there are too many outliers.The graph of the MSE loss function is shown in Fig.3.10. The general formula of MSE:

$$\text{Loss : MSE} = \frac{1}{N}(y_i - y')^2 \tag{3.7}$$

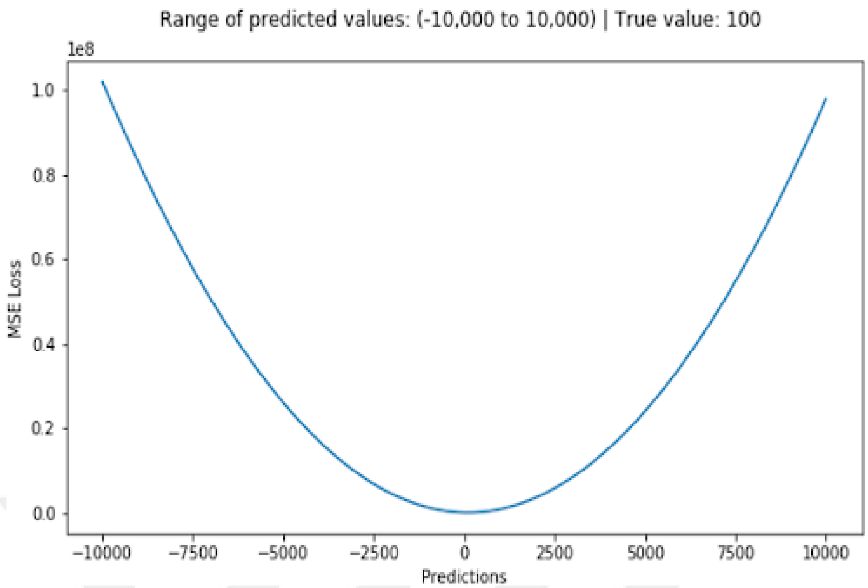


Figure 3.10 MSE Loss Function Graph [17]

3.4.2.2 Mean Absolute Error/L1 Loss:

The Mean Absolute Error(MAE) loss function calculates the average of the differences between the predicted and ground-truth value by taking the absolute value. It is the second most frequently used loss function among regression loss functions. MSE is much more robust to outliers because it does not calculate the square of large errors according to the loss function. Therefore, it is used if the data is prone to outliers. The graph of the MAE loss function is shown in Fig.3.11.

The general formula of MAE:

$$\text{Loss : MAE} = \frac{-|y_i - x_i|}{n} \tag{3.8}$$

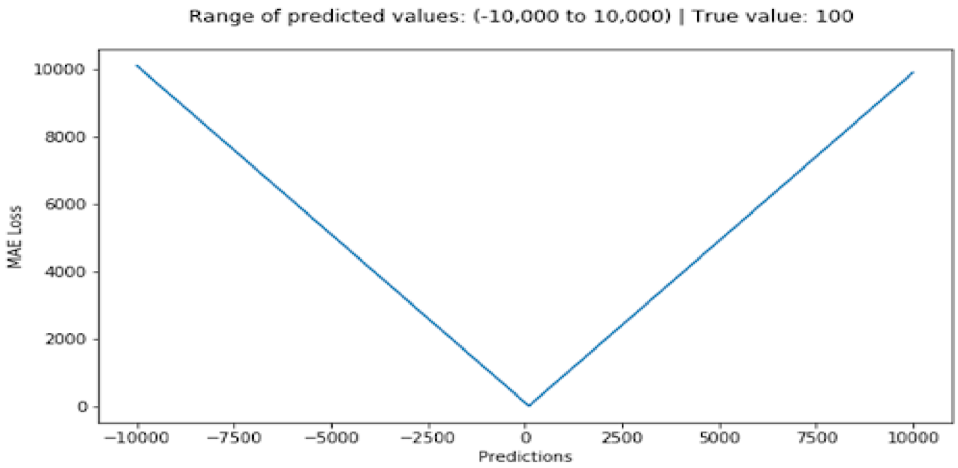


Figure 3.11 MAE Loss Function Graph [17].

3.4.3 Our Loss Function

The HomographyNet network, which we refer to for homography estimation, uses the classification loss function in eq.**3.4** for training since it uses a classification network in the last part of the network.In homography-based regression networks, the function in eq.**3.3** is generally used. Since we are using a regression network in the last part of our network, we need to use a regression loss function. The general formula of homography matrix is:

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \quad \textbf{(3.9)}$$

Since our study does not include rotation difference, rotation values(H_{11} , H_{12} , H_{21}, H_{22}) will be taken as 0. We also accept 1 as the scale(H_{33}). Therefore, while formulating the loss function, we actually only train according to the difference between the translation(H_{13}, H_{23}) values.Since we will only use translation values, only 2 parameters will be active in the 9-parameter homography matrix. Therefore, it will be sufficient to train our network according to the amount of error in these 2 parameters. Therefore, we train our network by slightly changing the L2, or mean square error, loss function. While training our network, we use the euclidean loss function (L2) at the end of the training process. We calculate the difference between the ground-truth matrix and the homography matrix produced by the network and square it to eliminate positive-negative. We apply this process to each pair of images in an iteration and collect the resulting errors. This gives us the loss amount of our network.The loss function is based on calculating the difference in translation values between the ground truth homography matrix and the homography matrix produced by our network for each image pair.The Euclidean loss function (L2) we use is expressed as follows.

$$\text{Loss : (L2)} = \sum_{i=1}^{n} \sqrt{(H_{13\_gt} - H_{13\_nt})^2 + (H_{23\_gt} - H_{23\_nt})^2} \qquad \textbf{(3.10)}$$

H_{13_gt}: Horizontal translation parameter for ground truth homography matrix.

H_{13_nt}:Horizontal translation parameter for the homography matrix produced by our network.

H_{23_gt}:Vertical translation parameter for ground truth homography matrix.

H_{23_nt}:Vertical translation parameter for the homography matrix produced by our network.

n: Total numbers of image pairs

Since there is no difference in rotation and scale, the simplified version of both the ground truth and the homography matrix estimated by the network is as follows:

$$H = \begin{bmatrix} 0 & 0 & H_{13} \\ 0 & 0 & H_{23} \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.11)}$$

3.2 DATASET PREPARATION

Image mosaicing studies in the literature stitch images with overlapping areas. Since mosacing has not been done before for non-overlapping images, we used plain image datasets with fewer edges and features to train the network more easily. So, we trained and tested our deep homography estimation algorithm with Kaggle's dataset containing landscape images [18]. This dataset include of mountain, desert, sea, island and beach landscape images(Fig. 3.12).

Figure 3.12 Some examples of Kaggle landscape image dataset [18].

While creating training, validation and testing set from Kaggle's landscape image datasets, we first resized all images to 1280x960 size. Then, from this large image, we created by randomly cutting 2 input images that have non-overlapping area, and stitch from the horizontal or vertical axis as in Fig. 3.13. Thus, a large dataset was created for image pairs that have non-overlap area. Our dataset contains 500000 image pairs for training and 5000 image pairs for validation and testing.

Figure 3.13 Some examples of image pairs that have stitch from horizontal and vertical axis respectively.The corner points of image A are shown in green, and the corner points of image B are shown in blue.

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 INTRODUCTION

We will examine the "Experiments and Results" phase in 3 stages. First, we explain the data set, processor, graphics card, etc. that we used when training our network in the "Implementation Details" section. Then, we talk about the experimental configuration section to examine the change of our network according to various parameters. Finally, we explain the visual and statistical results of our network in the "Results" section.

4.2 IMPLEMENTATION DETAILS

We trained our network for about 120 hours using the Intel Iris Xe Graphics GPU that using stochastic gradient descent (SGD) and momentum is 0.9. We basically set the learning rate to 0.005 and multiplied it to 0.1 after every 30,000 iterations. While the algorithm was trained with a total of 90,000 iterations, the batch size was chosen as 64.In our network, 500,000 image pairs were used as samples for training and 5000 for validation and testing phases as in Fig.4.1. While training our network from scratch, a model was created by the network as a result of the first 400 iterations, and this model is updated as a result of every 400 iterations during training. This model, which was created to prevent all processes from being wasted in the event of an unexpected event (for example, a sudden shutdown of the computer or an update situation) stopping the training of the network, is given as input to the system when the training code first runs. In addition, after the training is completed, this model is given as input to the software during the testing phase to test the learning algorithm.

```
C:\Users\HP\Desktop\Non-Overlap>python train.py
Found totally 500000 training files and 5000 validation files
start training
```

Figure 4.1 Start of the training network

When creating the dataset for training, we used Kaggle's landscape dataset [18], which includes desert, mountain, beach, island and sea images. Among these images, we cut

320x240 image pairs with non-overlapping areas from the large image according to the following algorithm:

1) A random point (P) is selected on the 1280x960 large Kaggle image and the coordinates of this point are named (u,v).This can also be called the coordinates of the starting point of the reference image.

2) The left, right, lower and upper regions of this P point are represented by a number from 1 to 4, respectively, and the dataset creation algorithm randomly selects a number from 1 to 4.This number determines in which region the target image will be relative to the reference image.

3) A value between -32 and +32 is chosen randomly. Lets define this in "rand". This value is also expressed as corner displacement value.The corner-displacement amount is the horizontal or vertical pixel difference between the points where the images stitch.In this way, we will calculate the coordinates (u',v') of the starting point (P') of the target image.

4) According to this algorithm, a relationship is formed between the target image (P') and reference image (P) starting points for each region as follows.

If the target image is stitched from the right side of the reference image:

$$P = (u, v)$$

$$P' = (u + 320, v + rand)$$

If the target image is stitched from the left side of the reference image: P=(u,v)

P'=(u-320,v+rand)

If the target image is stitched from the lower side of the reference image: P=(u,v)

P'=(u+rand,v+240)

If the target image is stitched from the upper side of the reference image: P=(u,v)

P'=(u+rand,v-240)

5) The difference between the vertical and horizontal coordinates of point P' and point P gives the amount of translation in the vertical and horizontal axes between the two images. This can be expressed by the homography matrix between two images.

Expressing this algorithm in pseudocode is as follows.

```
u ← RANDOM(0,1280); v
← RANDOM(0,960); rand
← RANDOM(−32,32);
corner ← RANDOM(1,4);
P ← (u,v) P′ ←
(u′,v′) if corner
= 1 then u′ =
u+320 v′ =
v+rand
else if corner = 2 then
u′ = u−320 v′ =
v+rand
else if corner = 3
then u′ = u+rand v′ =
v+240
else if corner =
4 then u′ =
u+rand v′ =
v−240 end if
```

Figure 4.2 Corner displacement between the image pairs. The green corner inside the red box is the upper left corner of the reference image. The blue corner is the lower left corner of the target image. The pixel difference between these 2 corners is defined as the corner-displacement.

For example, in Fig.4.2, green dots represent the corner points of the reference image and blue dots represent the corner points of the target image. The corner displacement value between the two images is defined as 20. Accordingly, there is a relationship between the starting points of the target image and the reference image as follows:

P=(u,v)

P'=(u-240,v-20)

According to this algorithm, target image and reference images are created from the large image in the Kaggle dataset. While these images are fed to our homography prediction network in Figure 7, they are converted into 128x128 grayscale images.

In this way, we have created 500,000 image pairs for training and 5000 image pairs for validation and testing.

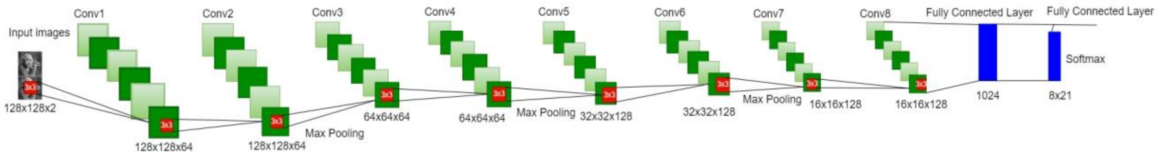


Figure 4.3 Our Network Architecture-2.

When creating image pairs that have non-overlap area from the horizontal or vertical axis, we define some corner-displacement amount between them. The corner-displacement amount is the horizontal or vertical pixel difference between the points where the images stitch. While creating the data set in our algorithm, this displacement amount was randomly selected to be a maximum of 32 pixels. For example, in the Fig.4.2, the corner-displacement amount for the image pairs whose corners are marked in green and blue is defined as the pixel difference on the horizontal axis between the green and blue corners in the red box. The corner-displacement amount for Fig.4.2 is approximately 20 pixels.

4.3 EXPERIMENTAL CONFIGURATION

In this section, we will explain the performance of our Deep Homography estimation network with different experiments. We will compare the performance of our algorithm on images with different corner-displacement amounts and different amounts of Gaussian noise, respectively.

4.3.1 Testing at Different Corner-Offset Amounts

In step 4.2, we explained that when testing our algorithm, we set a random corner offset value. In order to examine the variation of the performance of our network according to the amount of corner offset, we train our network by only changing the amount of corner offset, provided that the parameters such as GPU, number of iterations and training time remain constant.Below we describe the results when we train our mesh to have a maximum vertex offset of 32 and 96, respectively.

When measuring performance in the algorithm, we need to determine a metric. We calculate the median value of the pixel errors made in the horizontal and vertical axis for all image pairs in the test set. Thus, this median value gives us the average error value of the network in the horizontal and vertical axis for this data set.To measure the performance of our network, we take the median of the error values on the horizontal and vertical axes. In addition, we can observe the performance of our algorithm by graphing the average pixel errors on the horizontal and vertical axis. Table 4.1 shows the variation the median of pixel errors in the horizontal and vertical axes for the maximum corner-offset of 32 and 96 pixels.

Table 4.1 Error analysis in horizontal and vertical axis according to different corner-offset amounts.

X: Median of horizontal pixel errors.

Y: Median of vertical pixel errors.

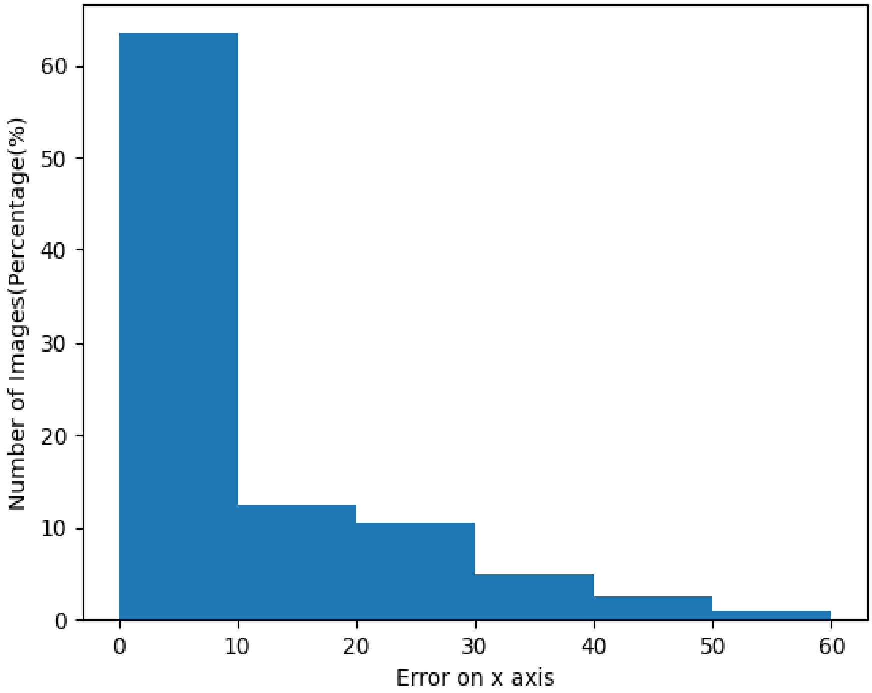
Max. Corner-displacement (pixel)	$\tilde{X}$ **(pixel)**	$\tilde{Y}$ **(pixel)**
32	1	1
96	**2**	**2**

While training our network, we first set the maximum random corner-offset value as 32 and trained our network accordingly for approximately 120 hours. First, we would like to give information about the performance of our network under these conditions. The graphs in Fig.4.4 show the performance results when we trained our network by setting the maximum corner-offset value to 32. The graphs show the distribution of pixel errors on the horizontal and vertical axis when we tested our data set, respectively. Accordingly, when we set the maximum corner-offset value to 32, the pixel error value occurring in more than 60 percent of our test set is less than 10.

Then, we increased the maximum corner-offset value to 96 and trained our network for 120 hours without changing the time parameter. The graphs in Fig.4.5 show the horizontal and vertical performance graphs of the test set when we trained our network with a maximum corner-offset value of 96 for 120 hours. Thus according to the graphics, when the maximum corner-offset value is 32, approximately 65 percent of the test set has an error rate of less than 10 pixels on the horizontal and vertical axis, but when we train our network to have a maximum corner-offset value is 96, this rate drops to approximately 55 percent on both axes.According to these tables, when we increase the maximum corner-offset value, we understand that we need to train our network more.

In the Fig.4.18 and 4.7 you can see examples of the visual results when we trained our network to have a maximum corner-offset value of 32.The corner points of reference image are shown in green, and the corner points of target image are shown in blue.

In the Fig.4.19 and 4.9 you can see examples of the visual results when we trained our network to have a maximum corner-offset value of 96.The corner points of reference image are shown in green, and the corner points of target image are shown in blue.



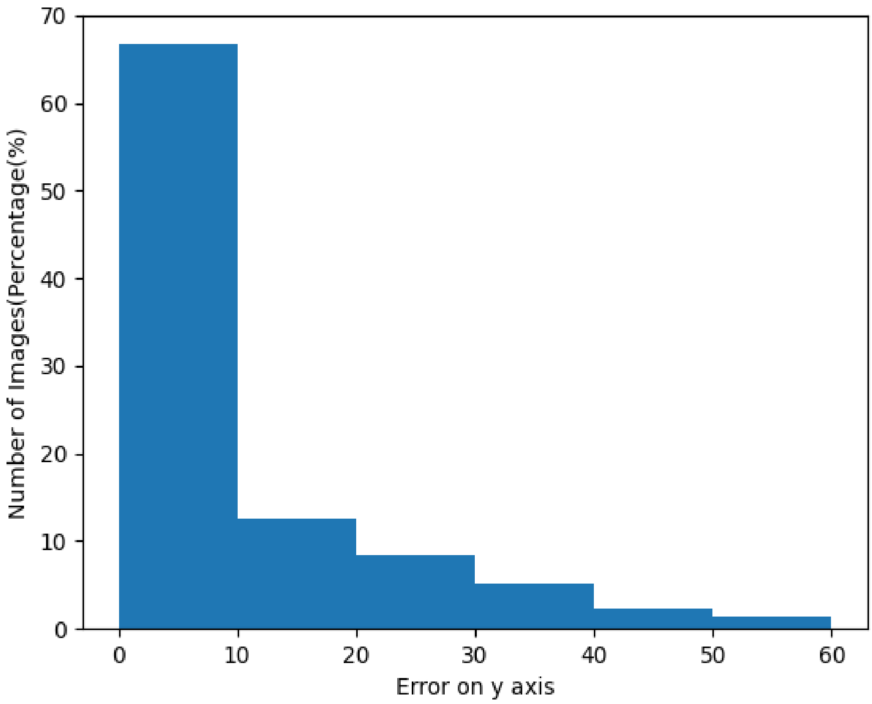


Figure 4.4 Percentage histogram graphs on horizontal and vertical axis respectively for maximum corner-offset of 32 pixels.

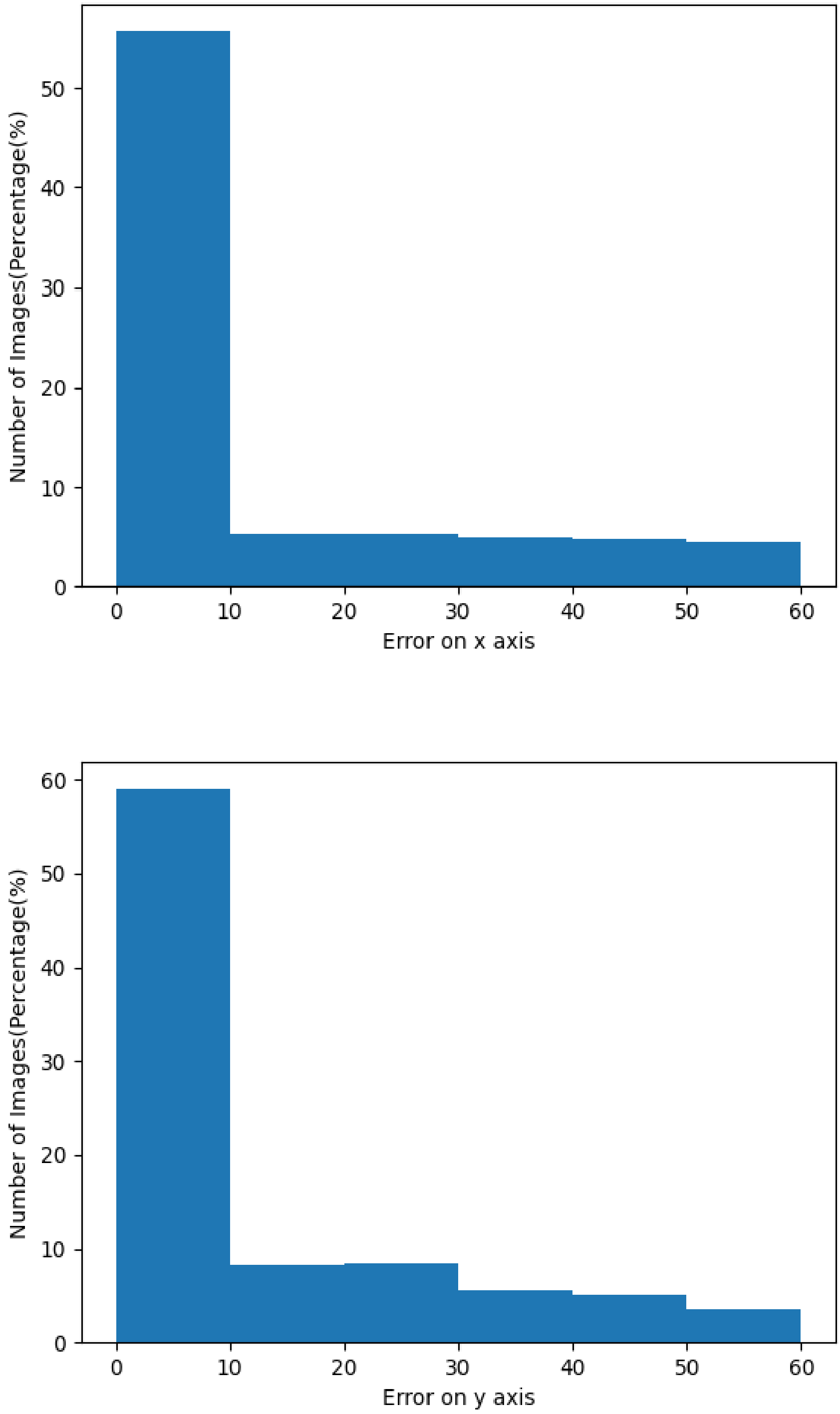


Figure 4.5 Percentage histogram graphs on horizontal and vertical axis respectively for maximum corner-offset of 96 pixels.

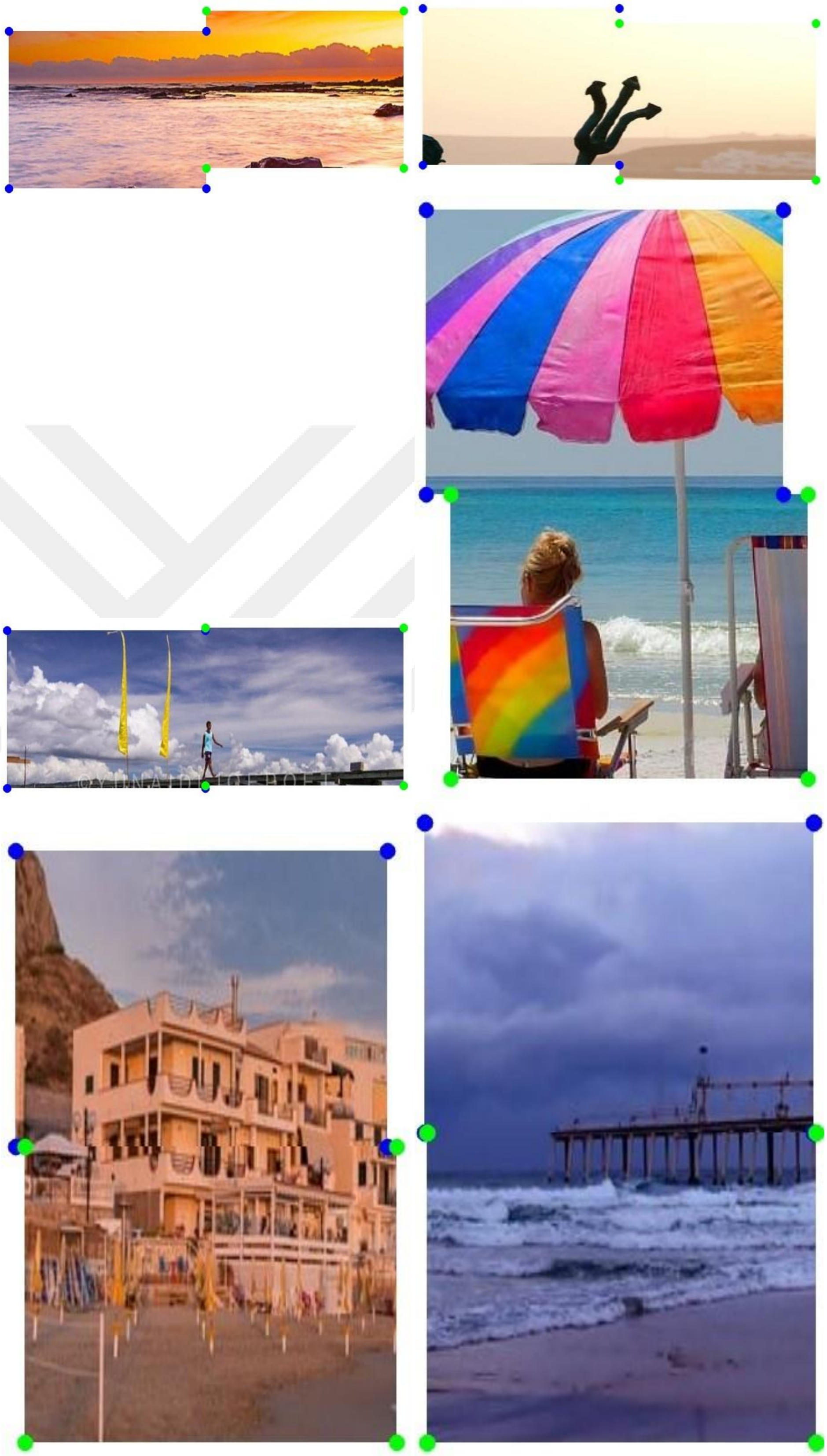


Figure 4.6 Stitching results for image pairs with corner-offsets up to 32 pixels.

Figure 4.7 Stitching results for image pairs with corner-offsets up to 32 pixels.

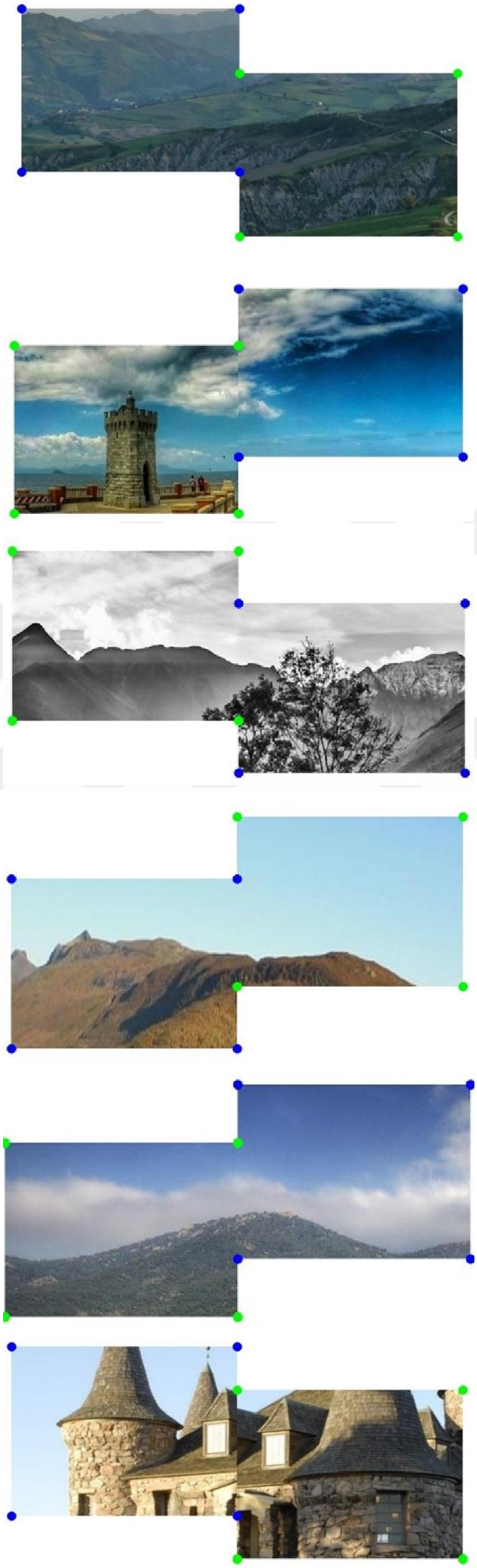


Figure 4.8 Stitching results for image pairs with corner-offsets up to 96 pixels.

Figure 4.9 Stitching results for image pairs with corner-offsets up to 96 pixels.

4.1.2 Testing at Different Amounts of Gaussian Noise

In this section, we tested our network with images containing different amount of Gaussian noise without changing other parameters such as maximum vertex-displacement amount, training time, GPU etc.In this section, we will first explain what Gaussian noise means in image processing. Then, we will explain the performance of our network on datasets containing Gaussian noise.

4.1.2.1 Gaussian Noise

In the field of image processing, randomly observed brightness or color changes on images are called image noise, and distortions in the image signal due to external sources are observed. The brighter area on the image, the more multiplicative noise that image is likely to contain. Image noises are similar to additives and can be modeled as follows:

A(x,y) = H(x,y) + B(x,y) The coordinates of the pixel to which noise is applied the horizontal and vertical axes;

A(x,y)= function of noisy image. H(x,y)= function of image noise. B(x,y)= function of noiseless image.

Fig.4.10 shows an example visual of the change that occurs when noise is added to original image.

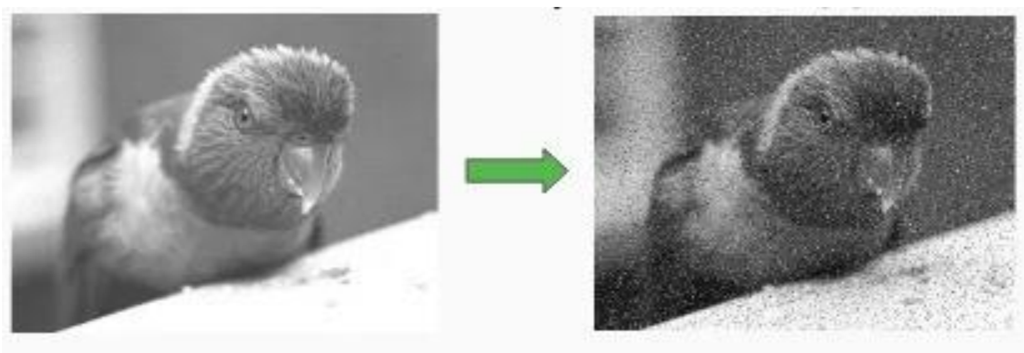


Figure 4.10 An example of image with noise [19].

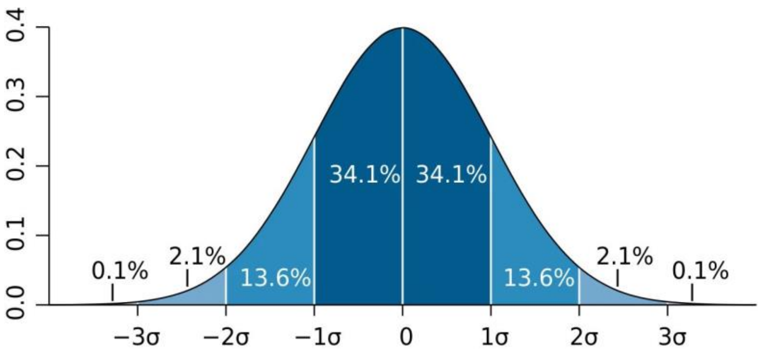


Figure 4.11 An example of a normal (Gaussian) distribution [20].

Statistical noise whose probability density function is equal to the normal distribution is called Gaussian noise.Gaussian Noise is also called Gaussian Distribution. Noise is created by randomly adding a Gaussian function to the image. It also occurs in amplifiers or detectors due to radiation of hot objects or thermal vibration of atoms. It can also be called electronic noise.Fig.4.11 shows an example of a Gaussian distribution.

Figure 4.12 An example of a image with no Gaussian noise. [20].

When gaussian noise is added to an image,the probability density function(PDF) p of a Gaussian random variable g is calculated by this formula:

$$P_G(g) = \frac{1}{\sigma 2\Pi} e^{\frac{-(g-u)^2}{2\sigma^2}} \qquad \textbf{(4.1)}$$

where, u represents the mean value, g is the gray level, and sigma is standard deviation. Accordingly, the magnitude of Gaussian noise varies in direct proportion to the sigma value. In machine learning-based image processing methods, it is aimed to make the models resistant to such noise by adding Gaussian noise to the images. The explanation of other parameters in the Gaussian noise formula is as follows:

Variance: Determines the amount of deviation of the noise. If deviation value is higher, the more noise the image contains.

Mean: Determines the average of the noise. If the average value is higher, the brighter the image.

Probability of applying transformation: Determines the probability of enlargement applied to the image. If Gaussian noise is applied to all images, the probability is 1.

Now we will examine the change that occurs when low and high variance noise is added to the image in Fig.4.12.

As you can see in Fig.4.13, as the amount of variance increases, the Gaussian noise contained in the image increases.

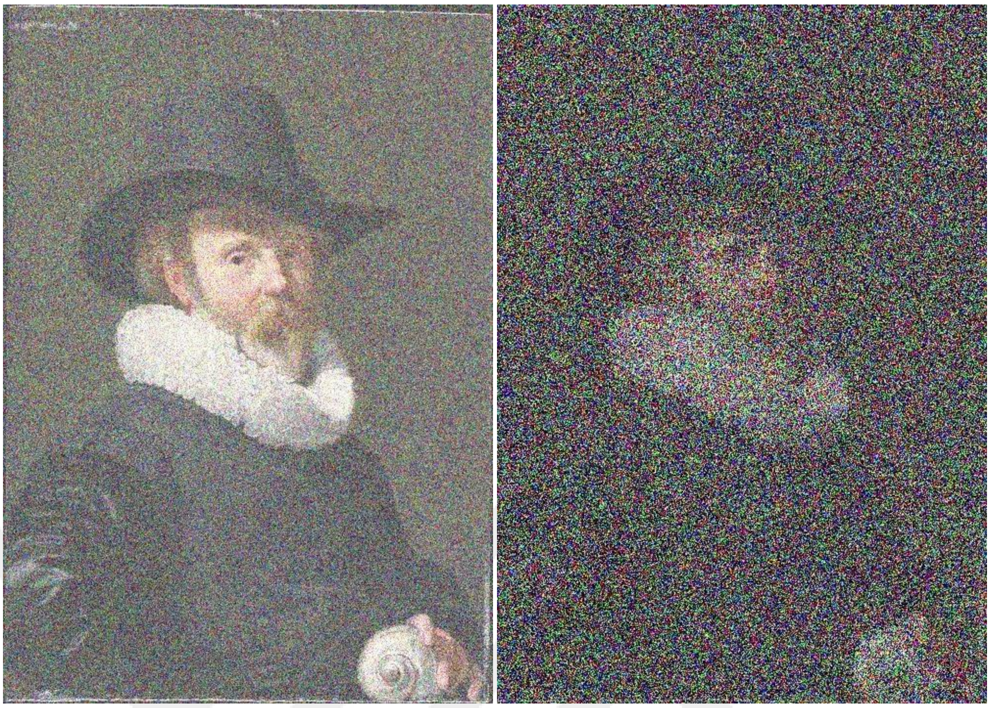


Figure 4.13 Images containing Gaussian noise with low and high variance, respectively [20].

4.1.3 Testing of Our Network with Different Amounts of Gaussian Noise

In this section, we tested our network with images containing different amount of Gaussian noise without changing other parameters such as maximum vertex-displacement amount, training time, GPU etc.

When measuring performance in the algorithm, we need to determine a metric. We calculate the median value of the pixel errors made in the horizontal and vertical axis for all image pairs in the test set. Thus, this median value gives us the average error value of the network in the horizontal and vertical axis for this data set. To measure the performance of our network, we take the median of the error values on the horizontal and vertical axes. In addition, we can observe the performance of our algorithm by graphing the average pixel errors on the horizontal and vertical axis. Table 4.2 shows the variation of the median of corner position errors in the horizontal and vertical axes for the different gaussian noise with different variances. As we can see from the table, the performance of the network decreases as the amount of gaussian noise increases.

Accordingly, if image pairs contain Gaussian noise, we need to train our network further.

Table 4.2 Error analysis in horizontal and vertical axis according to different gaussian-noise.

$\overline{X}$: Median of horizontal pixel errors.

Y: Median of vertical pixel errors.

Variance (pixel)	$\tilde{X}$ **(pixel)**	$\tilde{Y}$ **(pixel)**
0	1	1
0.001	45	20
0.005	83	43
0.1	**88**	**58**

In order to better visually observe the performance of image pairs containing different amounts of Gaussian Noise, we will visually explain the changes that occur in the results of the mosaicing process when Gaussian Noise is added to the image pairs containing the ground truth mosaicing results in Fig.4.14.We add Gaussian noise with a variance value of 0.001,0.005 and 0.1 to the image pairs in Fig.4.14 and show the results in Fig.4.15, 4.16, 4.17 respectively.

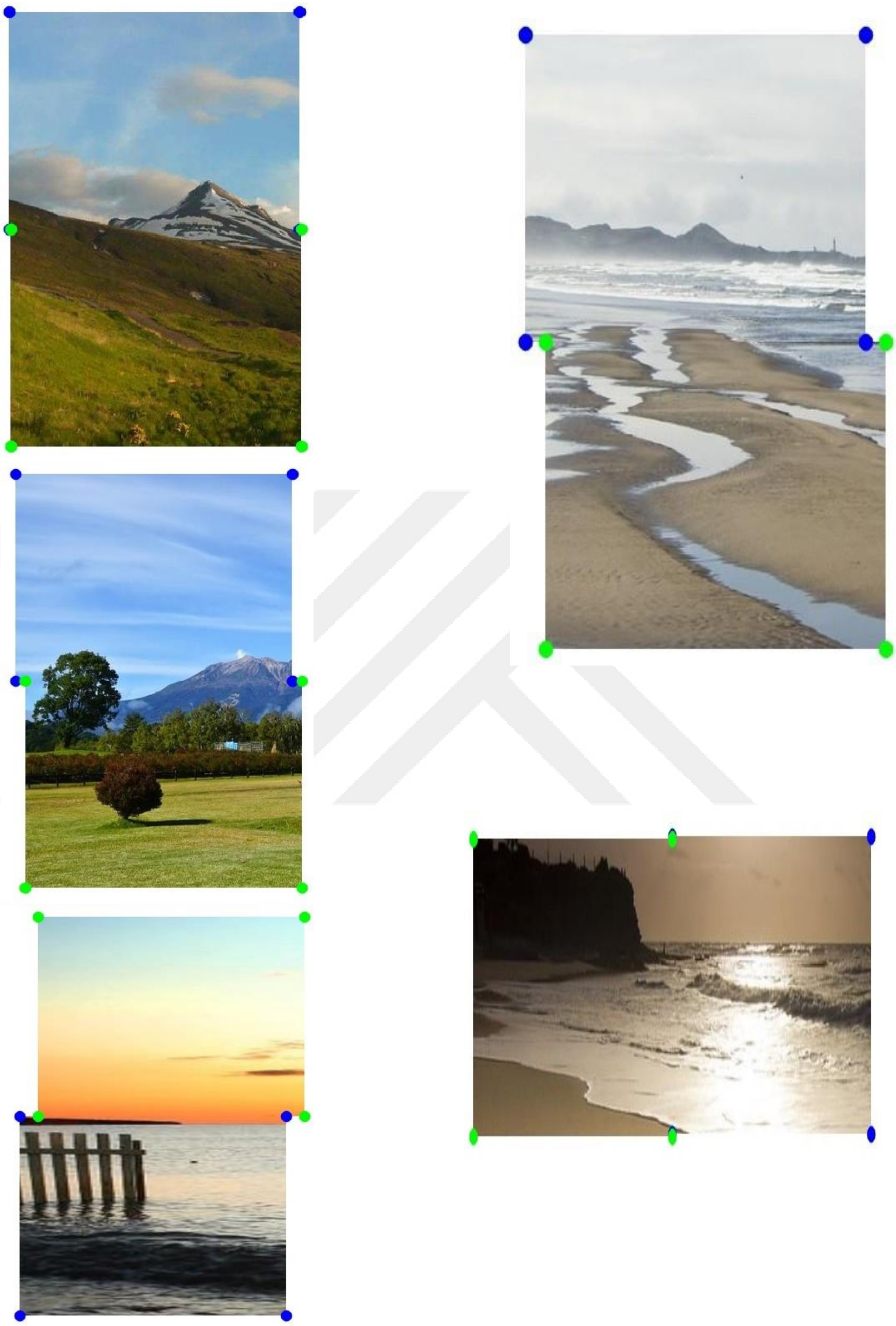


Figure 4.14 Ground truth stitching results for image pairs.The corner points of reference images are shown in green, and the corner points of target images are shown in blue.

Figure 4.15 Stitching results for image pairs with gaussian noise. Ground truth and variance values of 0.001 respectively. The corner points of reference images are shown in green, and the corner points of target images are shown in blue.

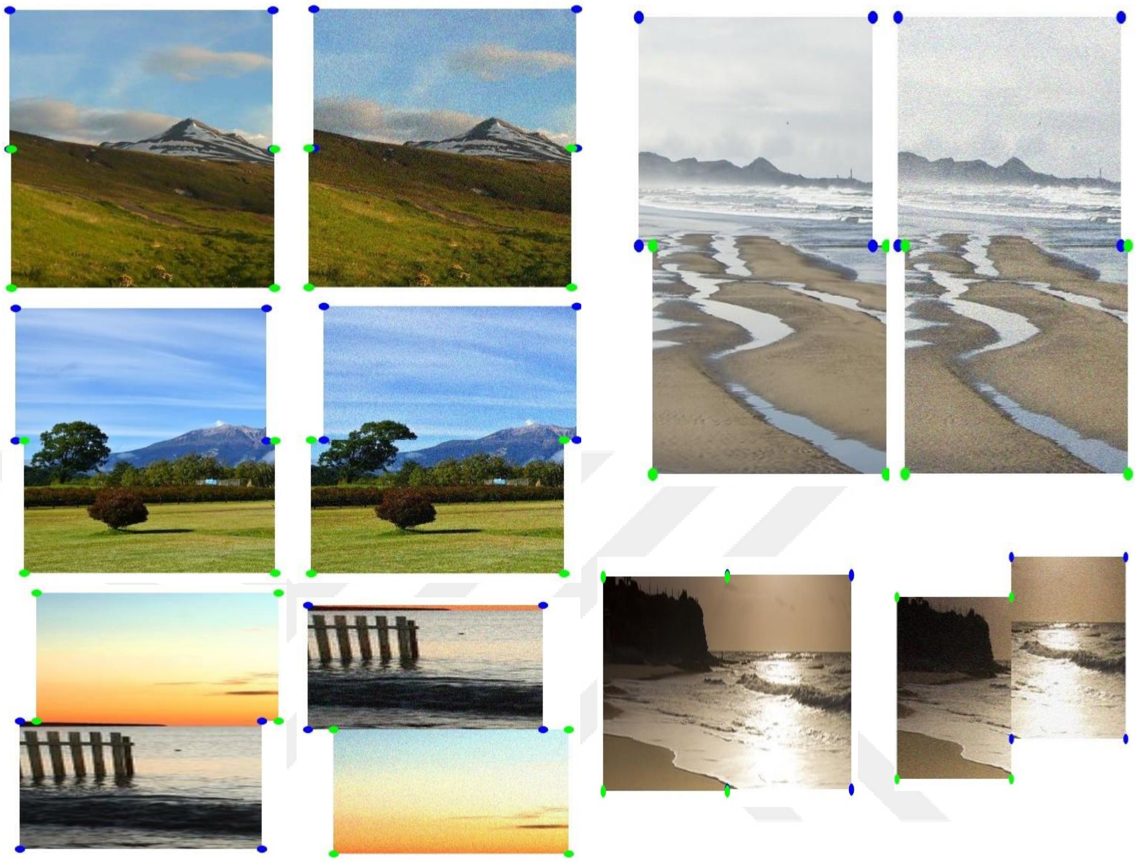


Figure 4.16 Stitching results for image pairs with gaussian noise. Ground truth and variance values of 0.005 respectively.The corner points of reference images are shown in green, and the corner points of target images are shown in blue.

Figure 4.17 Stitching results for image pairs with gaussian noise. Ground truth and variance values of 0.01 respectively.The corner points of reference images are shown in green, and the corner points of target images are shown in blue.

4.2 RESULTS AND COMMENTS

Our network has been tested for 5000 image pairs, ensure to mosaicing images that have non-overlap area. In the test process, we set the maximum corner-offset amount to 32 and our test set does not contain gaussian noise. As a result of this test applied to 320x240 image pairs, the median of the errors in the horizontal and vertical axes is 1 pixel. Contrary to the studies in the literature mosaicing overlapping images, our method yielded promising results in non-overlapping images.

In the previous step, we also explained the variation of the performance of our network according to the amount of corner-offset and gaussian noise. When we increase the maximum-corner offset amount to 96, the performance of our network has decreased a little and the median of the pixel errors in both axes is 2. We anticipate that this amount of error can be reduced by training our network a little more. As we explained in step .4.3.2, the performance of our homography estimation network is bad in images containing Gaussian noise. Below(Fig. 4.18, 4.19, 4.20) we present the visual results for all these experiments.

In Fig. 4.18 we show some visual results for pairs of images that do not contain gaussian noise and have offsets up to a maximum of 32 pixels. The visual results show that our method gives an overall promising result. Our method can sometimes make mistakes in pairs of images that contain too many or too few edges.

In Fig. 4.19, we have sampled some visual results for corner-offsets up to a maximum of 96 pixels. When the corner-offset amount increases, the error rate of our method has increased because it is necessary to train the network more.

Fig. 4.20 shows some visual results for image pairs containing gaussian noise. The images show, from left to right, the stitching results of pairs of images with ground truth, gaussian noise with variance of 0, 0.01, 0.05, and 0.1. According to the results, as the variance value increases, the amount of error increases.

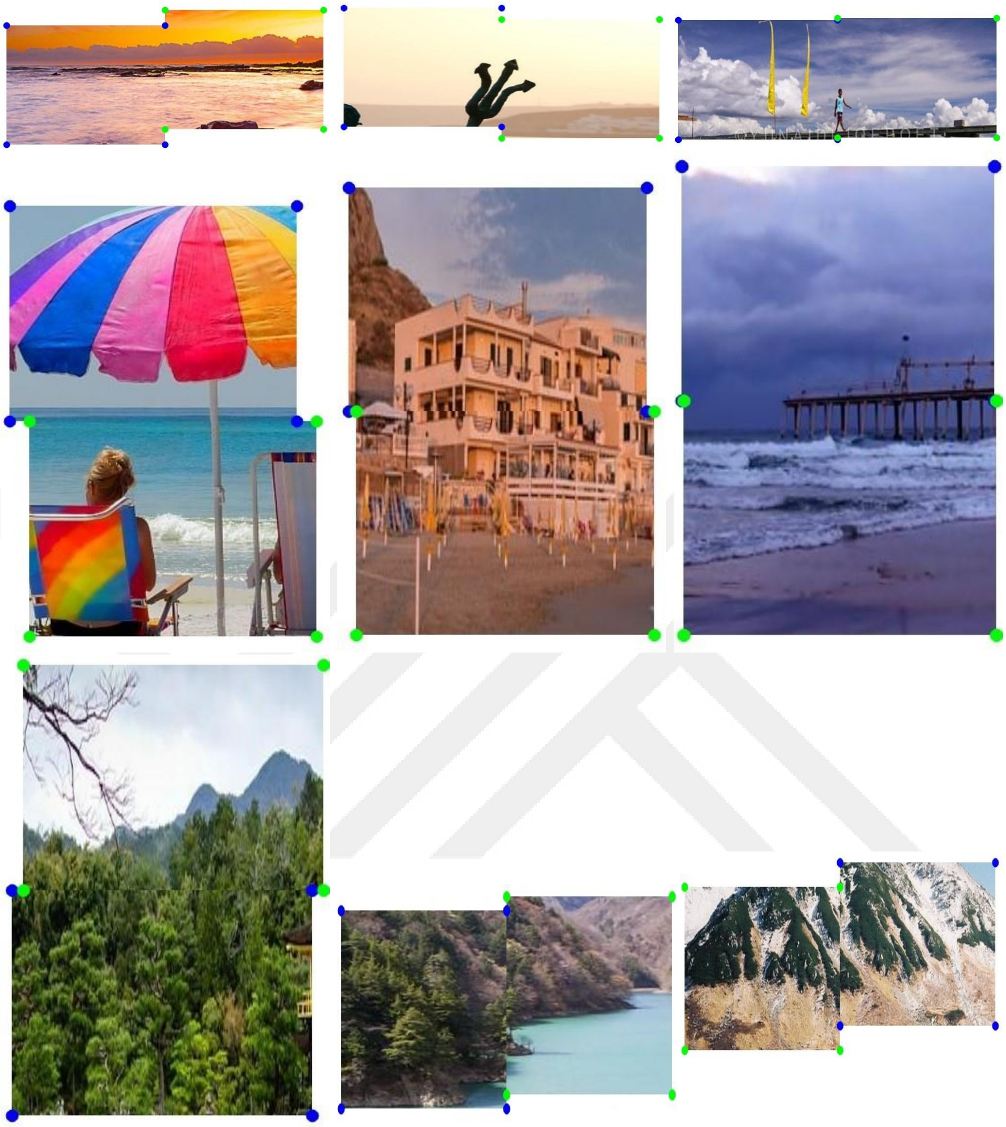


Figure 4.18 Stitching results for image pairs with corner-offsets up to 32 pixels and without noise. The corner points of reference images are shown in green, and the corner points of target images are shown in blue.

Figure 4.19 Stitching results for image pairs with corner-offsets up to 96 pixels and without noise. The corner points reference images are shown in green, and the corner points of target images are shown in blue.

Figure 4.20 Stitching results for image pairs with gaussian noise. Ground truth, variance values of 0.001 and 0.005 and 0.1, respectively. The corner points of reference images are shown in green, and the corner points of target images are shown in blue.

CHAPTER 5

CONCLUSION

In this thesis, we presented a deep learning based image mosaicing method. Unlike the studies in the literature that stitching images with overlapping areas, our method stitch images that have non-overlapping areas. We performed this process with a deep learning-based method and used a network that generates a homography matrix according to the amount of translation between image pairs that do not have rotation difference and scale change. We generated a custom dataset by clipping images from a Kaggle's landscape dataset. The network is trained and tested using this dataset under different scenarios including different displacement amounts and different noise levels. Our method showed a promising performance up to a large amount of displacements. However, the performance drops dramatically with increasing noise since there is no common regions in images and the weak features learned by the network are corrupted with noise. The performance of the method may be increased by training with a larger dataset including noisy images. In addition, since we train our network only on image pairs containing translation, our network also gives unsuccessful results on image pairs containing rotation. In order for our network to be successful on image pairs containing rotation, it must be trained with rotation.

5.1 LIMITATIONS OF THE STUDY

When we started this study, we expected that we would encounter difficulties because images with overlapping areas are generally mosaiced in the literature. When mosaicing images that have non-overlapping areas, we actually chose a computer vision-based method for the first time. However, since this method can be affected by many parameters such as brightness and exposure differences, we turned to learning-based methods. At this point, when we wanted to derive a relationship between two images, the most logical method for us was homography. However, we did not have a pre-trained model for mosaicing images that did not have overlapping areas. Therefore, to make our job easier, our first step was to eliminate the rotation and scale parameters. Then, we thought that using a very complex dataset would not be very efficient in terms of training the network, so we used a simple dataset containing landscape images from Kaggle. The biggest challenge we encountered was that since homography-based learning models included images with overlapping area, training of image pairs without overlapping area took a long time. So we had to train our network for days.

5.2 FUTURE WORK

Our study just mosaics a single non-overlapping image pair. The mosaicing of more than two images can be studied as a future work. For example, an application can be developped that allows randomly cutting 5 small images from a large image and then stitching them correctly again. We also aim to use rotation and scale information when creating a homography matrix between non-overlapping image pairs in our future studies. In addition, our network often shows bad results on datasets containing noise. In order to improve the performance of our network on the test set containing noise, we need to train our network with the data set containing noise. Our network gives promising results when trained for approximately 120 hours on a noise-free dataset. In order to show similar results on the noise-containing data set, our network must be trained with at least 120 hours of noise-containing image sets. In addition, since we train our network only on image pairs containing translation, our network also gives unsuccessful results on image pairs containing rotation.In order for our network to be successful on image pairs containing rotation, it must be trained with rotation.

REFERENCES

[1] Capel D., "Image mosaicing," in Image Mosaicing and super-resolution. Springer, 2004, pp. 47–79.

[2] Ghosh D. and Kaabouch N., "A survey on image mosaicing techniques," Journal of Visual Communication and Image Representation, vol. 34, pp. 1–11, 2016.

[3] DeTone D., Malisiewicz T., and Rabinovich A., "Deep image homography estimation," arrive preprint arXiv:1606.03798, 2016.

[4] Hays J., Brahmbbhatt S., and Lambert J., Local Feature Matching, 24 Sept 2018, https://faculty.cc.gatech.edu/~hays/compvision2018/proj2/.

[5] Joglekar J. and Gedam S. S., "Area based image matching methods—a survey," Int. J. Emerg. Technol. Adv. Eng, vol. 2, no. 1, pp. 130–136, 2012.

[6] Ghannam S. and Abbott A. L., "Cross correlation versus mutual information for image mosaicing," International Journal of advanced computer Science and applications (IJACSA), vol. 4, no. 11, 2013.

[7] HARRIS Corner Detection. 20 Jan 2024. Retrieved from https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_ harris.html

[8] FAST Algorithm for Corner Detection. 20 Jan 2024. Retrieved from https://docs.opencv.org/4.x/df/d0c/tutorial_py_fast.html

[9] Introduction to SIFT. 20 Jan 2024. Retrieved from https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

[10] Introduction to SURF. 20 Jan 2024. Retrieved from https://www.ccoderun.ca/programming/doxygen/opencv/ tutorial_py_surf_intro.html

[11] Contours : Getting Started. 20 Jan 2024.Retrieved from https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

[12] Sachdev A. Spatial and Frequency Domain – Image Processing. 2 Oct 2019.https://medium.com/vithelper/ spatial-and-frequency-domain-image-processing-83ffa3fc7cbc

[13] He B. and Yu S., "Parallax-robust surveillance video stitching," Sensors, vol. 16, no. 1, p. 7, 2015.

[14] Image Stitching. 20 Jan 2024. Retrieved from https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/

[15] Wang L., Yu W., and Li B., "Multi-scenes image stitching based on autonomous driving," in 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), vol. 1. IEEE, 2020, pp. 694–698.

[16] Shi Z., Li H., Cao Q., Ren H., and Fan B., "An image mosaic method based on convolutional neural network semantic features extraction," Journal of Signal Processing Systems, vol. 92, pp. 435–444, 2020.

[17] Common Loss Function. 20 Jan 2024. Retrieved from https://builtin.com/machine-learning/ common-loss-functions

[18] Kaggle Landscape Pictures. 20 Jan 2024. Retrieved from https://www.kaggle.com/datasets/arnaud58/ landscape-pictures

[19] Swain A. Noise in Digital Image Processing. 28 Aug 2018. https://medium.com/@anishaswain/noise-in-digital-image-processing-55357c9fab71#:~:text=Gaussian%20Noise% 20is%20a%20statistical,arises%20in%20amplifiers%20or%20detectors.

[20] GaussianNoise. 20 Jan 2024. Retrieved from https://wiki.cloudfactory.com/docs/mp-wiki/augmentations/ gaussian-noise

[21] Inampudi R. B., “Image mosaicing,” in IGARSS’98. Sensing and Managing the Environment. 1998 IEEE International Geoscience and Remote Sensing. Symposium Proceedings.(Cat. No. 98CH36174), vol. 5. IEEE, 1998, pp. 2363–2365.

[22] Ghosh D., Park S., Kaabouch N., and Semke W., “Quantitative evaluation of image mosaicing in multiple scene categories,” in 2012 IEEE International Conference on Electro/Information Technology. IEEE, 2012, pp. 1–6.

[23] Azzari P., Di Stefano L., and Mattoccia S., “An evaluation methodology for image mosaicing algorithms,” in Advanced Concepts for Intelligent Vision Systems: 10th International Conference, ACIVS 2008, Juan-les-Pins, France, October 20-24, 2008. Proceedings 10. Springer, 2008, pp. 89–100.

[24] Szeliski R. et al., “Image alignment and stitching: A tutorial,” Foundations and Trends® in Computer Graphics and Vision, vol. 2, no. 1, pp. 1–104, 2007.

[25] Nasibov A., Nasibov H., and Hacizade F., “Seamless image stitching algorithm using radiometric lens calibration for high resolution optical microscopy,” in 2009 Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control. IEEE, 2009, pp. 1–4.

[26] Berberidis K. and Karybali I., “A new efficient cross-correlation based image registration technique with improved performance,” in 2002 11th European Signal Processing Conference. IEEE, 2002, pp. 1–4.

[27] Miranda-Luna R., Daul C., Blondel W. C., Hernandez-Mier Y., Wolf D., and Guillemin F., “Mosaicing of bladder endoscopic image sequences: Distortion calibration and registration algorithm,” IEEE Transactions on Biomedical Engineering, vol. 55, no. 2, pp. 541–553, 2008.

[28] Dame A. and Marchand E., “Video mosaicing using a mutual information-based motion estimation process,” in 2011 18th IEEE International Conference on Image Processing. IEEE, 2011, pp. 1493–1496.

[29] Jain D. K., Saxena G., and Singh V. K., "Image mosaicing using corner techniques," in 2012 International Conference on Communication Systems and Network Technologies. IEEE, 2012, pp. 79–84.

[30] Levin A., Zomet A., Peleg S., and Weiss Y., "Seamless image stitching in the gradient domain," in Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV 8. Springer, 2004, pp. 377–389.

[31] Xiong Y. and Turkowski K., "Registration, calibration and blending in creating high quality panoramas," in Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98 (Cat. No. 98EX201). IEEE, 1998, pp. 69–74.

[32] Xiong Y., "Eliminating ghosting artifacts for panoramic images," in 2009 11th IEEE International Symposium on Multimedia. IEEE, 2009, pp. 432–437.

[33] Szeliski R., Uyttendaele M., and Steedly D., "Fast poisson blending using multi-splines," in 2011 IEEE International Conference on Computational Photography (ICCP). IEEE, 2011, pp. 1–8.

[34] El-Saban M., Izz M., Kaheel A., and Refaat M., "Improved optimal seam selection blending for fast video stitching of videos captured from freely moving devices," in 2011 18th IEEE International Conference on Image Processing. IEEE, 2011, pp. 1481–1484.

[35] Gracias N., Mahoor M., Negahdaripour S., and Gleason A., "Fast image blending using watersheds and graph cuts," Image and Vision Computing, vol. 27, no. 5, pp. 597–607, 2009.

[36] Wen H. and Zhou J., "An improved algorithm for image mosaic," in 2008 International Symposium on Information Science and Engineering, vol. 1. IEEE, 2008, pp. 497–500.

[37] Nie L., Lin C., Liao K., Liu S., and Zhao Y., "Unsupervised deep image stitching: Reconstructing stitched features to images," IEEE Transactions on Image Processing, vol. 30, pp. 6184–6197, 2021.

[38] Lai W.-S., Gallo O., Gu J., Sun D., Yang M.-H., and Kautz J., "Video stitching for linear camera arrays," arXiv preprint arXiv:1907.13622, 2019.

[39] Li J., Zhao Y., Ye W., Yu K., and Ge S., "Attentive deep stitching and quality assessment for omnidirectional images," IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 1, pp. 209–221, 2019.

[40] Hoang V.-D., Tran D.-P., Nhu N. G., Pham T.-A., and Pham V.-H., "Deep feature extraction for panoramic image stitching," in Intelligent Information and Database Systems: 12th Asian Conference, ACIIDS 2020, Phuket, Thailand, March 23–26, 2020, Proceedings, Part II 12. Springer, 2020, pp. 141–151.

[41] Nie L., Lin C., Liao K., and Zhao Y., "Learning edge-preserved image stitching from large-baseline deep homography," arXiv preprint arXiv:2012.06194, 2020.

[42] Nie L., Lin C., Liao K., Liu M., and Zhao Y., "A view-free image stitching network based on global homography," Journal of Visual Communication and Image Representation, vol. 73, p. 102950, 2020.

[43] Simonyan K. and Zisserman A., "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[44] Nguyen T., Chen S. W., Shivakumar S. S., Taylor C. J., and Kumar V., "Unsupervised deep homography: A fast and robust homography estimation model," IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2346–2353, 2018.

[45] Zhang J., Wang C., Liu S., Jia L., Ye N., Wang J., Zhou J., and Sun J., "Content-aware unsupervised deep homography estimation," in Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. Springer, 2020, pp. 653–669.

[46] Le H., Liu F., Zhang S., and Agarwala A., "Deep homography estimation for dynamic scenes," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 7652–7661.

[47] Joshi H. and Sinha K., “A survey on image mosaicing techniques,” International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 2, no. 2, pp. 365–369, 2013.

[48] Baker S., Datta A., and Kanade T., “Parameterizing homographies,” Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-06-11, 2006.

[49] Hartley R. and Zisserman A., Multiple view geometry in computer vision. Cambridge university press, 2003.

[50] ImageNet. 20 Jan 2024. Retrieved from https://www.image-net.org/

[51] AlexNet. 20 Jan 2024. Retrieved from https://www.analyticsvidhya.com/blog/2021/03/ introduction-to-the-architecture-of-alexnet/