Exploring Relationships Between Ground and Aerial Views by Synthesis and Matching

Krishna Regmi

University of Central Florida

Part of the Computer Sciences Commons

Find similar works at: https://stars.library.ucf.edu/etd2020

University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2020- by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation
https://stars.library.ucf.edu/etd2020/747
EXPLORING RELATIONSHIPS BETWEEN GROUND AND AERIAL VIEWS
BY SYNTHESIS AND MATCHING

by

KRISHNA REGMI
M.S. Southern Illinois University Edwardsville, 2015
B.S. Tribhuvan University, 2009

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2021

Major Professor: Mubarak Shah
ABSTRACT

Cross-view images, referring to the images taken from aerial and street views, contain drastically differing representations of the same scene of a given location. Due to the differences in the camera viewpoints of ground and aerial images the same semantic concepts in the two viewpoints look very different. Therefore the problem of relating them is very challenging. Thus, it becomes crucial to explore the cross-view relations and learn appropriate representations such that images from these two domains can be associated.

In this dissertation we explore the relationship between ground and aerial views by synthesis and matching. First, we explore supervised approaches for cross view image synthesis problem to generate realistic images from the target (eg. ground) view, given an image from a source (eg. aerial) view. We solve this problem by utilizing Generative Adversarial Networks (GANs) to synthesize the target images and an auxiliary output, the target view segmentation maps, from source view images. We do so by enforcing the networks to correctly align and orient the different semantics in the scene by jointly penalizing the networks on the quality of target view images and the semantic segmentation maps. Next, we explore the geometrical cues between the aerial and ground images and attempt to preserve the pixels from aerial images to synthesize the ground images. We use homography to transform the aerial images to the street-view and preserve the pixels from the overlapping field of view, followed by inpainting the remaining regions in the ground image. Geometrically transformed images as input ease the network’s burden in synthesizing the cross-view images. Following the cross-view image synthesis problem, we solve the cross-view image matching problem. We propose a novel framework that uses the synthesized images for bridging the domain gap between the images from the two (aerial and ground) viewpoints and helps to learn better features for the cross-view image matching. Finally, the last part of the dissertation addresses the problem of matching the frames of a video with geo-tagged reference images for purpose of
geo-localization. We develop a novel method that learns coherent features for individual frames in
the query video by attending to all the frames of the video. We conduct extensive evaluations to
validate that the proposed approach performs better compared to methods that learn image features
independently.
EXTENDED ABSTRACT

Exploring the relationships between the aerial and ground images and videos for the tasks of synthesis and retrieval is very challenging due to the differences in the viewpoints these two set of data are captured from and limited overlap in the field of view (FOV) between them. An image from aerial view captures a wide area of a scene with roofs of buildings, roads, and trees from a camera orthogonal to the ground. A corresponding ground level image captures areas with the building facades, trees and sidewalks as well as perspective view of roads with heavy occlusions between the objects. In this dissertation, we explore the cross-view relations and learn appropriate representations such that these can be associated for image synthesis and retrieval.

In chapter 3, we explore supervised approach for cross view image synthesis problem to generate realistic images from the target (eg. ground) view, given an image from a source (eg. aerial) view. We solve this problem by utilizing Generative Adversarial Networks (GANs) to synthesize the target images and an auxiliary output, the target view segmentation maps, from source view images. We do so by enforcing the networks to correctly align and orient the different semantics in the scene by jointly penalizing the networks on the quality of target view images and the semantic segmentation maps. We conduct extensive qualitative and quantitative evaluations on Dayton and CVUSA datasets to validate the effectiveness of our methods compared to the baselines.

Next, in chapter 4, we propose a novel approach to perform geometrically-guided cross-view image synthesis. We leverage the geometrical cues between the aerial and ground images and attempt to preserve the pixels from aerial images to synthesize the ground images. We use homography to transform the aerial images to the street-view and preserve the pixels from the overlapping field of view, followed by inpainting the remaining regions in the ground image. Geometrically transformed images as input ease the network’s burden in synthesizing the cross-view images. We
conduct extensive evaluations on SVA dataset and demonstrate the superiority of our approach over baselines and previous methods in terms of qualitative and quantitative evaluations. As a result, the synthesized images are much sharper than purely learning based method used in previous chapter and are able to retain object structures and road boundaries in the target view.

While cross-view image synthesis is about generating new images from a different viewpoint, in chapter 5, we solve the cross-view image matching problem. Here, we find the matching (most similar) image for a query image by computing its feature similarity with the images in the gallery. We propose a novel framework that uses the synthesized images for bridging the domain gap between the images from the two (aerial and ground) viewpoints and helps to learn better features for the cross-view images. These learned features are next employed to solve cross-view geo-localization. Our extensive experiments show that the proposed joint feature learning method outperforms the state-of-the-art methods on CVUSA dataset and with feature fusion, we obtain significant improvements on top-1 and top-10 retrieval accuracies. Furthermore, we evaluate the generalization of the proposed method for urban landscapes on our newly collected cross-view localization dataset with geo-reference information.

In the Chapters 3 to 5, we utilize image datasets and work on cross-view images. Finally, in Chapter 6, we address the video geo-localization problem where we find the matching image for the frames of a video by comparing the frame features with the features of the geo-tagged reference images. We develop a novel method that learns temporally and geographically coherent features for individual frames in the query video by attending to all the frames of the video. Once the GPS location for each frame in the query video is estimated, its possible that there exists some outliers in the set of estimated GPS values. So, we propose a deep learning approach to trajectory smoothing by predicting the outliers in the estimated GPS positions and learning the offsets to smooth the trajectory. We benchmark a new dataset for the problem of video geo-localization that consists of videos from four different regions of the USA. We conduct extensive evaluations to validate that the
proposed approach performs better compared to methods that learn image features independently and our method generalizes well to different regions of the USA.
ACKNOWLEDGMENTS

I would like to extend my deepest appreciation to my Ph.D. advisor Dr. Mubarak Shah for all of his support and guidance over the last several years. Also, I would like to thank committee members, Dr. Ladislau Boloni, Dr. Yogesh Singh Rawat and Dr. Scott Branting for accepting to be a part of my committee and providing helpful guidance during the process of proposing and defending my dissertation. And, I would like to extend regards to Dr. Ali Borji for his mentorship during the early years of my PhD.

I would also like to thank my co-authors for their thoughtful discussions during our collaborations. Also, I would like to thank all of the past and present members of Center of Research in Computer Vision (CRCV), Cherry Place, Tonya LaPraire, Dr. Chen Chen, Dr. Yonatan Tariku Tesfaye, Dr. Leulseged Tesfaye Alemu, Dr. Dong Zhang, Dr. Haroon Idrees, Dr. Nasim Souly, Dr. Gonzalo Vaca, Dr. Khurram Soomro, Dr. Sarfaraz Hussein, Dr. Waqas Sultani, Dr. Shervin Ardeshir, Dr. Mahdi Kalayeh, Dr. Yicong Tian, Dr. Amir Mazaheri, Dr. Aidean Sharghi, Dr. Mohhamed Elfeki, Dr. Rodney Lalonde, Aisha Orooj Khan, Praveen Tirupattur, Aayush Rana, Muhammad Junaid Khan, Kevin Duarte, Robert Browning, Jyoti Kini, Sarah Shiraz, Ugrur Demir, Mamshad Rizve, Ishan Dave, Rohit Gupta, Swetha Sirnam, Madeline Schiappa, Maliha Arif, Alina Ageichik, Bruce McIntosh, Sandesh Sharma, Fawad Ahmed Keyani for valuable discussions, laughter and great memories.

Lastly and most importantly, I would like to extend my thanks to all who loved and supported me during this journey of pursuing this degree. To my wife, Heema Poudel, thank you so much for your endless support during the highs and the lows of my Ph.D; you have always been on my side. To my grandmother, parents and my brother, thank you so much for support and encouragement for pursuing my studies. And to all my friends not named in this document, I have been blessed to have you all in my life and I would not be here without you all.
# TABLE OF CONTENTS

LIST OF FIGURES ................................................................. xiv

LIST OF TABLES ................................................................. xxii

CHAPTER 1: INTRODUCTION ....................................................... 1

1.1 Cross View Image Synthesis Using Conditional Generative Adversarial Networks 2

1.2 Geometry-Guided Cross View Image Synthesis ........................................... 3

1.3 Cross View Image Matching ............................................................... 5

1.4 Video Geo-Localization Employing Geo-Temporal Feature Learning and GPS Trajectory Smoothing ...................................................... 8

1.5 Organization ........................................................................... 10

CHAPTER 2: LITERATURE REVIEW ........................................... 12

2.1 GANs ............................................................................. 12

2.2 Image Synthesis ................................................................ 13

2.3 Image Matching .................................................................. 15

2.4 Video Localization and Trajectory Smoothing .............................. 16

2.5 Geo-localization Datasets ....................................................... 17
CHAPTER 3: CROSS VIEW IMAGE SYNTHESIS USING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS .......................... 20

3.1 Methodology ................................................................. 20

3.1.1 Crossview Fork (X-Fork) Network .................................. 21

3.1.2 Crossview Sequential (X-Seq) Network ............................ 24

3.2 Experimental Settings ..................................................... 26

3.2.1 Dataset ................................................................. 26

3.2.2 Network and Implementation Details ............................... 27

3.3 Results ................................................................. 29

3.3.1 Qualitative Evaluation ............................................... 30

3.3.2 Quantitative Evaluation ............................................. 35

3.3.3 Generated Segmentation Maps .................................... 43

3.3.4 Visualization of Quantitative Evaluation (Scatterplot) ......... 46

3.3.5 kNN ................................................................. 47

3.4 Summary ................................................................. 48
5.3 Results ................................................................. 73
  5.3.1 Evaluation Metrics .............................................. 73
  5.3.2 Ground-to-Aerial Image Matching: Results on CVUSA Dataset ........ 73
  5.3.3 Aerial-to-Ground Image Matching: Reverse Problem ................. 81
  5.3.4 Cross-view Localization on UCF-OP Dataset ........................ 82

5.4 Summary ............................................................ 87

CHAPTER 6: VIDEO GEO-LOCALIZATION EMPLOYING GEO-TEMPORAL FEATURE
  LEARNING AND GPS TRAJECTORY SMOOTHING ........................... 88

6.1 Methodology ......................................................... 89
  6.1.1 Geo-Temporal Feature Learning Network (GTFL) ................. 89
  6.1.2 Loss Functions .................................................. 91
  6.1.3 Trajectory Smoothing Network .................................. 95

6.2 Experimental Setup ................................................ 97
  6.2.1 Datasets ......................................................... 97
  6.2.2 Implementation Details ........................................ 99

6.3 Results ............................................................. 100
  6.3.1 Evaluation Metrics ............................................ 100
LIST OF FIGURES

1.1 Example images in overhead/aerial view (left) and ground-level/street-view (right). We use conditional generative adversarial networks (cGANs) to solve the task of cross-view image synthesis. ................................................................. 2

1.2 An aerial image $I_a$ shown in the left is first transformed to street-view perspective using homography ($I_{ah}$). The transformed image needs inpainting in upper box region ($R_1$) and further processing in car region ($R_2$). These two regions are generated and the region in between is copied from homography transformed image $I_{ah}$ to obtain $I_g'$. We further train $I_g'$ to smooth the region boundaries and add realism to it. $I_g$ is the corresponding ground truth image. ................................................................. 4

1.3 For a given query ground panorama, the task is to find the matching aerial image from the gallery set. Here, we propose to first synthesize an aerial image for the query panorama using a generator network. Then, we use the synthesized aerial image together with ground panorama to obtain the aggregated query feature and use it for matching with gallery image features. The synthesized aerial image helps to bridge the domain gap between the query and gallery images and ease the cross-view matching task. ................................................................. 6

1.4 Given the video on the left panel (top), we extract its frames as shown on the left panel (bottom). For each frame, we determine its GPS location and plot on the aerial image as shown on the right panel. The path connecting these locations forms the GPS trajectory that the vehicle takes in the video. ................................................................. 8
3.1 X-Fork architecture. The Generator Network G forks at the penultimate layer to synthesize two output images, ground level RGB image and ground level semantic segmentation map for an input aerial image.  

3.2 Generator network of the X-Fork architecture shown in Figure 3.1. BN means batch-normalization layer. The first six blocks of decoder share weights, forking at the penultimate block. The number of channels in each convolution layer are shown below each blocks.  

3.3 X-Sequence architecture. The first generator (G1) synthesizes the ground level image for input aerial image and the second generator (G2) takes the output of G1 as its input and synthesizes the ground segmentation map. The two GANs, GAN-1 and GAN-2 work in sequence and thus it is called X-sequence architecture.  

3.4 Original ground and aerial image pairs (first row) from the training set and the images with segmentation masks from pre-trained RefineNet [43] overlaid on original images (second row).  

3.5 Example images generated by different methods in lower ($64 \times 64$) resolution in $a2g$ (aerial-to-ground) and $g2a$ (ground-to-aerial) directions.  

3.6 Example images generated by different methods in higher ($256 \times 256$) resolution in $a2g$ (aerial-to-ground) and $g2a$ (ground-to-aerial) directions.  

3.7 Qualitative results for Aerial to Ground level ($a2g$) synthesis using different methods on Dayton dataset. The networks learn to generate road, sidewalk (rows 2, 5, 7, 8), trees, houses (rows 1, 3, 5, 7, 8), pole (rows 2, 3, 4, 5) in the images.
3.8 Qualitative results for Ground to Aerial (g2a) synthesis using different methods on Dayton dataset. ................................................................. 34

3.9 Qualitative results of our methods and baselines on CVUSA dataset in a2g direction. First two columns show true image pairs, next four columns show images generated by Zhai et al. [102], Pix2pix[33], X-Fork and X-Seq methods, respectively. ........ 36

3.10 Qualitative results for Aerial to Ground level (a2g) synthesis using different methods on CVUSA dataset. ................................................................. 37

3.11 Segmentation overlay: Original image pairs are shown in first column, second column shows segmentation maps obtained from RefineNet [43] overlaid on original images; these segmentation maps are used as ground truth segmentation maps. The next two columns show segmentation maps from our proposed X-Fork and X-Seq methods overlaid on original images. The color representation of common semantic classes from Dayton dataset are provided below the images. .................. 43

3.12 Scatterplot of SSIM scores obtained using different methods (left: X-Fork vs Pix2pix and right: X-Seq vs Pix2pix) for test images of Dayton dataset. Images for different points in the scatterplot are shown. .................. 46

3.13 The first two columns show aerial and ground image pairs, third column shows corresponding street-view image synthesized by X-Seq method and next three columns show nearest images in the training set retrieved by computing L_1 distance between generated ground image and training set images. .............................. 47
4.1 An aerial image $I_a$ shown in (a) is first transformed to street-view perspective using homography ($I_{ah}$), shown in (b). The transformed image, $I_{ah}$ needs inpainting in upper box region ($R_1$) and further processing in car region ($R_2$), as shown in (c). These two regions are generated and the region in between is copied from homography transformed image $I_{ah}$ to obtain $I_g'$. We further train $I_g'$ to smooth the region boundaries and add realism to it. $I_g$ is the corresponding ground truth image (e).

4.2 Example images generated by different methods in aerial-to-ground (a2g) direction for SVA dataset. Upper panel: First column shows the aerial input images, followed by their homography transformed images in second column. Column 3-6 show images synthesized by different methods using the aerial images directly as input, i.e. without homography transformation. Lower panel: Column 1-5 shows images synthesized by different methods by utilizing homography transformed images as input. Finally, the last column shows the ground truth street-view images.

5.1 Joint Feature Learning: Inputs to this network are $I_g$ and $I_a$ and outputs are $f_g$ and $f_a$. Employing auxiliary loss between $f_a'$ and $f_a$ helps to pull features $f_g$ and $f_a$ closer, and minimize the domain gap between two features than when training two-stream network on ($I_g$, $I_a$) pairs. The branch in the middle (dotted box filled with cyan) is used during the training only.

5.2 Feature Fusion. Inputs to this network are $I_g$ and $I_a$ and outputs are $f_g^*$ and $f_a^*$. $f_g$ and $f_a$ are concatenated and passed through fully-connected layer (FC) to get their fused representation $f_g^*$.
5.3 Image retrieval examples on CVUSA dataset [102]. For each query ground-level panorama, the synthesized aerial image is shown alongside, followed by the five closest aerial images retrieved by proposed Feature Fusion method. The correct matching (ground truth) aerial images are shown in green boxes. Rows 5 and 6 show examples where the ground truth aerial images are retrieved at the second and fourth positions respectively. ............................................. 70

5.4 Comparison of different versions of our methods with CVM-Net I and CVM-Net II [30] on CVUSA dataset [102]. ................................................................. 76

5.5 T-SNE [50] visualization of aerial and ground image features obtained using the two-stream baseline (left) and the proposed feature fusion method (right) for 100 test images on the CVUSA dataset [102]. ............................................. 77

5.6 Some examples of failure cases. The numbers on the right show the position where the ground-truth aerial images were retrieved. ............................................. 80

5.7 Comparison of our methods with the baselines on CVUSA dataset [102] for reverse problem of aerial-to-ground image matching. ............................................. 82

5.8 Qualitative Results on CVUSA dataset [102] for aerial-to-ground image matching. Images with green borders are the ground-truth panoramas for the corresponding query images. ................................................................. 83

5.9 Image retrieval examples on the OP dataset. The correct aerial image matches are shown in green borders. The numbers below each aerial image shows its distance in meters from query ground image. ......................................................... 84

5.10 Geo-localization results on the OP dataset with different error thresholds. ........... 85
Cross-view image retrieval examples on the OP Dataset. Ground-truth aerial images are shown in green boxes. The number below each aerial image is its distance in meters from the query image. The first three rows present the images from Orlando and the next three rows of images are from Pittsburgh.

Geo-Temporal Feature Learning (GTFL) Network: Given a set of \( n \) frames for BDD (query) and corresponding \( n \) frames for GSV (gallery), frame embeddings are obtained using Encoders (VGG-16 network). Then, the Geo-Temporal Attention and Geo-Attention modules learn coherent feature representations for BDD and GSV frame embeddings respectively. The frame-based features are aggregated using Max-pooling operation to obtain the representative clip features, \( b_{clip} \) and \( g_{clip} \). The Frame Triplet loss and GPS loss are applied on frame level features, and the Clip Triplet loss is applied on clip features. Detailed architecture for Attention module is shown on the right where BDD/GSV features are learnt for the frame embeddings.

Scatterplot showing the relationship between feature distances (\( d_{feats} \)) and geographical distances (\( d_{gps} \)) for a subset of the dataset. Each point in the plot represents the feature distance between two images along x-axis and their geographical distance along y-axis. The blue line is the line of best fit through the scatterplot and shows a linear relationship can be established between the points; with a slope of 1.077 and intercept of -0.2313. We model the GPS loss to preserve the linear relationship between the feature distances and gps distances.
6.3 Proposed Trajectory Smoothing Network: The noisy GPS sequence \([GPS_1, GPS_2, \ldots, GPS_n]\) is input to the network to compute the error offsets \([\Delta GPS_1, \Delta GPS_2, \ldots, \Delta GPS_n]\) and the confidence scores \([p_1, p_2, \ldots, p_n]\) for each input value. The offsets are added to only those GPS values in the input sequence if they are deemed to be noisy by their confidence scores.

6.4 The frames from a BDD sequences at times \(T = 3, 6, 9, 12, 15\) and \(41\) (Left Panel, Top); and GSV images corresponding to the same GPS locations (Left Panel, Bottom). The plot of the trajectory (green curve) along with the frame locations (red and blue dots) for different times on the aerial map (Right Panel). Numbers 3, 6, 9, 12, 15 and 41 marked with blue dots on aerial map illustrate the position of moving camera at respective times.

6.5 Comparison of the proposed approach with baseline 2D-CNN in terms of recall accuracies.

6.6 Qualitative Results. Example Images showing the ground truth (green curves) and predicted (red curves) trajectories for Bay Area (first row) and San Francisco (second row).

6.7 Qualitative Results. Example Images showing the ground truth (green curves) and predicted (red curves) trajectories for New York (first row) and Berkeley (second row).

6.8 Trajectory smoothing example for a query clip from Berkeley region. (a) shows the ground truth (green curves) and the predicted trajectories before smoothing (blue curves). (b) show the ground truth (green curves) and the predicted smooth trajectories (red curves).
T-SNE [50] plot showing BDD and GSV features obtained for baseline 2D CNN (left plot) and proposed method (right plot) on test images. The plots show clusters representing the feature spaces for frames belonging to the same clip; which is more obvious in the right plot.
### LIST OF TABLES

3.1 Network Details of encoder and decoder used in GANs. Different operations carried out at each block are shown along with output dimensions. *Conv:* Convolution, *lReLU:* Leaky ReLU, *BN:* Batch Normalization, *ReLU:* Rectified Linear Unit, *UpConv:* Upconvolution. ........................................... 28

3.2 KL divergence scores between conditional and marginal probabilities (Inception Score). ......................................................... 38

3.3 Accuracies: Top-1 and Top-5. .................................................. 39

3.4 KL Divergence between model and data distributions. .................. 40

3.5 SSIM, PSNR and Sharpness Difference(SD) between real data and samples generated using different methods. .......................... 41

3.6 Frechet Inception Distance (FID) scores on Dayton and CVUSA datasets. .......................... 42

3.7 Evaluation Scores (per-class accuracy and mean Intersection over union (mIOU)) for segmentation maps generated by X-Fork and X-Seq methods for aerial-to-ground (a2g) and ground-to-aerial (g2a) synthesis. The scores .......................... 45

4.1 % of user preferences over images synthesized by different methods (over 100 images from the SVA dataset). .......................... 61

4.2 Quantitative evaluation of samples generated using different methods on SVA Dataset in a2g direction. IS - Inception Score, Acc. - Accuracy .......................... 62
5.1 Comparison of Top-1, Top-10 and Top-1% recall for the baselines and the proposed approaches (first panel) and with previous methods (second panel) on CVUSA Dataset [102].

5.2 Ablation Study on CVUSA Dataset [102]. The reported numbers are the retrieval accuracies for Feature Fusion network for specified ablation criteria.

5.3 Image matching performance in terms of Top-1, Top-10 and Top-1% recall on CVUSA Dataset [102] for aerial-to-ground matching.

5.4 Top-1 retrieval accuracy on Orlando-Pittsburgh Dataset.

6.1 The GPS window of each region under consideration and the area of each region in square kilometers and the number of clips considered from each of the regions from BDD dataset. We employ video sequences only from San Francisco area for training and video sequences from all four areas for testing as shown in the last column of this table.

6.2 Comparison of proposed approach with baseline methods in terms of localization error (meters). 2D CNN : Evaluation using raw features from pretrained VGG network. 2D CNNf : VGG network fine-tuned on our dataset.

6.3 Ablation study of GPS errors in meters with respect to different losses during the training. TLf : Frame Triplet loss; TLc : Clip Triplet loss; GL : GPS Loss.

6.4 Ablation on the number of heads in self-attention module and the number of transformer encoder layers.
6.5 Ablation study on varying feature dimension size. We report the evaluations in terms of localization error in meters. 108

6.6 Ablation study on experiments with and without NetVLAD layer in the proposed network. We report the evaluations in terms of geo-localization error in meters. 108
CHAPTER 1: INTRODUCTION

With the availability of more aerial and ground level data, and the access to image pairs from the two views, there has been a growing interest in learning the view independent features of the images for better understanding of different regions of the earth. Aerial and satellite imageries provide us with an ability to study the earth from above while the ground level images help understand the scene from close. Though there exists a large difference in the viewpoints, the aerial and the ground images captured at a given location represent the same scene and thus the same semantic meaning. Each domain (aerial and ground) has been studied separately by the computer vision community for different tasks in the past. For ground level images, some of these tasks include image geo-localization [24, 92, 101], semantic segmentation [13], building damage assessment [57], etc. Land cover and land use [79, 85]; anthropogenic changes [14, 40], changes due to natural disasters like hurricane [42], earthquakes [94]; monitoring cultural heritages [15] have been explored in aerial images. Some of the computer vision problems can be better solved if there is a mechanism to relate these two domains and study them jointly.

In this dissertation, we explore the relationships between these two domains in context of different computer vision tasks, such as synthesis, retrieval and geo-localization. We synthesize ground level image from an aerial image and vice versa. For retrieval, we learn view independent features for the images to find the best matching images in the gallery for a query image. Once a query image can be matched with geo-tagged reference image, its geo-localization can be determined. We discuss the motivations, challenges and our proposed solution to tackle each problem next.
1.1 Cross View Image Synthesis Using Conditional Generative Adversarial Networks

Here, we take the first step towards relating the aerial and ground images, which is to synthesize a ground level image from an aerial image and vice versa. Figure 1.1 illustrates some example images in the two views. The images reflect the great diversity and semantic richness in two views, illustrating the challenging nature of cross-view image synthesis task.

![Aerial Images](image1)

**Figure 1.1**: Example images in overhead/aerial view (left) and ground-level/street-view (right). We use conditional generative adversarial networks (cGANs) to solve the task of cross-view image synthesis.

View synthesis is a long-standing problem in computer vision. Previous works in view synthesis [19, 106, 81] dealt with generation of images containing single objects or natural scenes with very little variation in viewing angles between the source and target images. The network learns to copy large parts of image content from the input. The synthesis task is more challenging when views are drastically different, fields of views have little or no overlap, and objects are occluded. With increase in the number of objects in the scene and the increase in viewpoint differences, the synthesis task becomes more challenging. This is due to the increase in underlying factors that contribute to the variations (e.g., occlusions, shadows, etc). Thus, the network needs to learn that the corresponding images in each view need to contain all details and place them in correct positions with proper spatial orientations.
In this work, we attempt to solve the problem of synthesizing ground-level images from overhead imagery and vice versa using conditional Generative Adversarial Networks [54]. We consider images that are from two drastically different views, have small fields of view overlap, and some objects in the images may be occluded. As a result, learning to map the pixels between the views is difficult since the corresponding pixels in two views may represent different object classes. To address this challenge, we propose to use the semantic segmentation maps of target view images to regularize the training process. We do so by designing the network architecture that synthesizes cross-view segmentation map as an auxiliary output of the network along with the cross-view image. By encouraging the networks to generate the segmentation maps in target view, the network learns the semantic classes of each pixel which provide important cues in generating cross-view images. We propose two new cGAN architectures that generate images as well as segmentation maps in target view. The first architecture, called X-Fork, forks at the penultimate block to generate two outputs, target view image and segmentation map. The second architecture, called X-Seq, consists of stack of two connected GANs. The target view image generated by the first GAN is fed to the second GAN to generate its corresponding segmentation map. Once trained, both architectures are able to generate better images than the baseline that learns to generate the cross-view images only. This implies that learning to generate segmentation map along with the target image indeed improves the quality of generated image. Details of this work are covered in chapter 3.

1.2 Geometry-Guided Cross View Image Synthesis

While the synthesis of cross-view images using an end-to-end generative adversarial networks can work well as explained in previous section, we believe that learning to preserve the pixel values from the source to the target view helps to ease the burden of the network in solving such
a challenging task of cross-view image synthesis. However, simply copying and pasting of pixels from one view to another would not be a solution when there exists a large difference in viewpoints between the source and the target views. Rather, it is important to learn the semantic classes present in the input view and how they would orient themselves in the target view, i.e their transformation between the source and the target views.

![Figure 1.2](image.png)

**Figure 1.2:** An aerial image $I_a$ shown in the left is first transformed to street-view perspective using homography ($I_{ah}$). The transformed image needs inpainting in upper box region (R1) and further processing in car region (R2). These two regions are generated and the region in between is copied from homography transformed image $I_{ah}$ to obtain $I_{g'}$. We further train $I_{g'}$ to smooth the region boundaries and add realism to it. $I_g$ is the corresponding ground truth image.

Here, we propose to exploit the geometric relationship between the ground and aerial views to guide the synthesis. For this, we first compute the homography transformation matrix between the views and then project the aerial images to street-view perspective. By doing so, we obtain an intermediate (homography transformed) image that looks very close to the target view image, however not as realistic and contains some missing regions (Image $I_{ah}$ in Figure 1.2). Now, our problem reduces to preserving the scene layout and details in $I_{ah}$ while filling in the missing regions and adding realism to the image. For this, we use cGAN architectures, X-Fork and X-Seq, described in chapter 3 and call them H-Fork and H-Seq, since the inputs are the homography transformed images. Additionally, we propose a method to preserve details from the homography transformed image in a controlled setting to generate the ground view images. Overall method constitutes of three subtasks: a) generating missing regions (R1 in Figure 1.2) by inpainting, b) adding realism to
the details preserved from source to target view (R2 in Figure 1.2), and c) blending the regions R1 and R2 with the remaining pixels of the homography transformed image. We use different cGANs that work specifically for each subtask. We call this approach H-Regions. Details of this work are provided in chapter 4.

1.3 Cross View Image Matching

In the earlier sections, we focused on relating the cross-view images for the task of image synthesis. Here, we relate the aerial and ground images for the task of image matching or retrieval. More specifically, given a query image in aerial view, we want to retrieve the ground level image belonging to the same location, from a reference database of ground level images. Assuming, we know the GPS location of the images in the reference dataset, we can assign the GPS of the matching ground image to the query aerial image.

Traditionally, for estimating the geo-location of an image the matching has been conducted between images taken from the same view, primarily street-view [24, 70, 100], which have a high degree of visual similarity in terms of scene contents. Since these ground level reference images are typically concentrated around urban areas with more human accessibility, the applicability of these methods is limited to those regions. With the availability of aerial images from Google maps, Bing maps, etc. that cover the earth surface densely, researchers have lately explored the prospect of cross-view image matching [30, 45, 92], where the query ground image is matched against aerial images. This comes with additional challenges due to variation in viewpoints between the ground and aerial images, which capture the same scene differently in two views. This motivates us to explore transforming the query street-view image into aerial view, so that the transformed image has scene representations similar to the images it is matched against.
Figure 1.3: For a given query ground panorama, the task is to find the matching aerial image from the gallery set. Here, we propose to first synthesize an aerial image for the query panorama using a generator network. Then, we use the synthesized aerial image together with ground panorama to obtain the aggregated query feature and use it for matching with gallery image features. The synthesized aerial image helps to bridge the domain gap between the query and gallery images and ease the cross-view matching task.

The recent success of Generative Adversarial Networks (GANs) [21] in synthesizing realistic images from randomly sampled noise vectors [63] or conditional variables such as text [64, 103], images [33], labels [54], etc. and also based on our work covered in chapters 3-4 have inspired us to frame the problem as viewpoint translation followed by feature matching. Moreover, GANs have been used for domain transfer problems as in [36, 107] to learn the mapping between different
domain representations. Recent cross-view synthesis works by [65, 66, 17, 108] are successful in transforming the images between aerial and street views. In chapter 5, we address the following problem: given a ground-level panorama, retrieve the matching aerial image. In order to solve this problem, we take a next step to synthesize aerial images from ground-level panorama and use them for image retrieval.

The complexity of the cross-view image synthesis problem and its challenges are well-known. Thus, the synthesized images cannot be relied on to completely replace the query ground-level image to solve the matching task. Therefore, we propose a framework as shown in Figure 1.3 to incorporate the synthesized image into the matching pipeline as auxiliary information in order to bridge the existing domain gap between aerial and ground view images. We attempt to learn feature representations for aerial reference images that are similar to their corresponding ground level images, as well as the synthesized aerial images. Since the synthesized aerial images are transformed representations of street-view (ground) images, we expect them to contain representative features. By learning representations in this manner, the synthesized aerial images force the network to minimize the distance between feature representations of aerial images and street-view images. Additionally, we hypothesize that some features of aerial images are better learned by considering synthesized aerial images rather than street-view images. Thus, the joint training of these image triads (ground, synthesize aerial from ground, and corresponding real aerial) will help the aerial stream retain important cues that would have otherwise been lost in cross-view training. We fuse the learned complementary feature representations of synthesized images with query image features to obtain a robust representation that we use for our image matching task. Details of this work is presented in chapter 5.

1 street-view and ground view will be used interchangeably
1.4 Video Geo-Localization Employing Geo-Temporal Feature Learning and GPS Trajectory Smoothing

In this final chapter, we extend the idea of image matching to videos. More specifically, given a query video recorded from a moving camera, we want to determine the GPS locations of each frame in the video. The set of GPS locations for the query frames represents the geo-trajectory for the moving camera. The predicted trajectory may not always be smooth due to some outliers in the retrieved set, thus we propose a deep learning based approach to smooth the trajectory.

Figure 1.4: Given the video on the left panel (top), we extract its frames as shown on the left panel (bottom). For each frame, we determine its GPS location and plot on the aerial map as shown on the right panel. The path connecting these locations forms the GPS trajectory that the vehicle takes in the video.

Figure 1.4 shows a video on the top of the left panel and below it are the frames at different time instances. Given the video, we extract the frames as shown and determine the geo-location of each frame. We plot these geo-locations on the aerial map as shown on the right panel; and connect the locations to visualize the path that the vehicle is following in the video. In this work, we explore the task of video geo-localization on the same-view data, where both query videos and gallery images
are from the ground view. One possible way to solve this problem is to treat each frame in the video independently and apply frame-based matching methods to determine the GPS location of each frame. Earlier works in video geo-localization [22, 88] are based on classical computer vision methods, where first the SIFT descriptors [49] are computed for each frame in the clip as well as for the images in the gallery set (reference database). Then, for each frame in the query video the best matching reference image is computed and its corresponding GPS location is assigned to the query frame and the predicted GPS trajectory is obtained. Recent works [26, 31, 101] use 2D CNN networks to obtain frame-level features instead of SIFT features for query frames and the gallery images and follow the same approach for image matching. The features for each frame in the clip are expressed independent to each other and thus predicted GPS locations may not be smooth enough to represent a realistic trajectory of the moving camera since no temporal closeness between the frames is exploited directly while learning the features for the clips. In this work, we propose to leverage the geo-temporal proximity between the video frames while learning their features, in order to enforce the predicted locations of the consecutive video frames to be close to each other.

Motivated by the recent success of deep learning methods in video understanding and the effectiveness of transformer networks [89] to incorporate long range context dependencies between the inputs, we propose to use transformer based architecture to learn feature representations for the video frames of the query videos. The network captures the coherent features for the video frames and hence provides smoother predicted trajectories for the query videos. In addition to exploiting the temporal proximity between the frames within a clip, we propose to use a novel GPS loss to learn smoother features for clips that are geographically closer to each other. Typical imagery captures areas containing vegetation, landmarks and landscapes unique to those areas and can extend over a small geographical region. So, the clips over this geographical region should share similar feature representations. Thus, we propose to learn similar features for video clips corresponding
to the same geographical locations by constraining the training of our proposed network by using GPS loss. Once the GPS locations for the query video is estimated, earlier works used b-spline [22] and minimum spanning tree based trajectory reconstruction algorithms [88] to smooth the initial estimates of these GPS positions. Here, in this work, we propose a transformer encoder based trajectory smoothing network to determine the noisy GPS outliers in a set of estimated GPS locations for the query clip and smooth the GPS value if it is determined to be noisy by the network.

There is no publicly available large-scale dataset for video geo-localization to evaluate the capability of the proposed framework. The dataset of Vaca-Castano et al. [88] consists of only 45 query videos making it impractical to train deep learning methods. Heng et al. [26] use dataset with image pairs that do not fit to our problem formulation. So, in this work, we build a new video geo-localization benchmark dataset by utilizing Berkeley Driving Dataset (BDD) videos [97] and by collecting matching Google StreetView (GSV) images. The BDD videos are the query videos and the GSV images constitute the gallery set. The dataset covers four different regions of the USA; San Francisco, Berkeley, Bay Area and New York. We evaluate on query videos from all four regions. Details of this effort is provided in chapter 6.

1.5 Organization

In Chapter 2, we review the existing literature on Generative Adversarial Networks for image synthesis, as well the related works in image matching for the task of geo-localization between the cross-view images. In Chapter 3, we present our proposed cross-view image synthesis methods using generative adversarial networks. In Chapter 4, we explain our geometry-based cross-view synthesis network that attempts to preserve the geometrical relationships between the cross-view images. In Chapter 5, we propose a GAN-based network for bridging the cross-view domains to facilitate the cross-view image matching. Finally, in Chapter 6, we solve the task of video geo-
localization by learning geographically and temporally coherent features for query video frames using the proposed transformer-based feature learning network; followed by smoothing of the estimated geo-trajectory.
CHAPTER 2: LITERATURE REVIEW

In this chapter, we conduct the literature review of the topics related to this dissertation. We start by reviewing the GAN literature. This is followed by the review of research works on image synthesis, image matching and video localization and trajectory smoothing. We also explore the existing geo-localization datasets on images and videos. Finally we review the applications of transformer networks and feature aggregation in deep neural networks.

2.1 GANs

Generative Adversarial Network (GAN) architecture [21] consists of two adversarial networks: a generator and a discriminator that are trained simultaneously based on the min-max game theory. The generator $G$ is optimized to map a $d$-dimensional noise vector (usually $d=100$) to an image (i.e., synthesizing) that is close to the true data distribution. The discriminator $D$, on the other hand, is optimized to accurately distinguish between the synthesized images coming from the generator and the real images from the true data distribution. The objective function of such a network is:

$$\min_G \max_D L_{GAN}(G,D) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log (1 - D(G(z)))] \tag{2.1}$$

where, $x$ is real data sampled from data distribution $p_{data}$ and $z$ is a $d$-dimensional noise vector sampled from a Gaussian distribution $p_z$.

Conditional GANs synthesize images employing some auxiliary variable which may be labels [54], text embeddings [103, 64] or images [33, 107, 36]. In conditional GANs, both the discriminator and the generator networks receive the conditioning variable represented by $c$ in Equation (2.2).
The generator uses this additional information during image synthesis, while the discriminator makes its decision by looking at the pair of conditioning variable and the image it receives. The real pair input to the discriminator consists of true image from real data distribution and its corresponding label, while the fake pair consists of the synthesized image and the label. For conditional GAN, the objective function is:

$$\min_G \max_D L_{cGAN}(G, D) = E_{x,c \sim p_{data}(x,c)}[\log D(x, c)] + E_{x',c \sim p_{data}(x',c)}[\log (1 - D(x', c))], \tag{2.2}$$

where $x' = G(z, c)$ is the generated image.

In addition to the GAN loss shown in Equation (2.2), previous works (e.g., [33, 107, 61]) have tried to minimize the $L_1$ or $L_2$ distances between real and generated image pairs (commonly known as the reconstruction loss). This step aids the generator to synthesize images very similar to the ground truth. Minimizing $L_1$ distance generates less blurred images than minimizing the $L_2$ distance. That is, using the $L_1$ distance increases image sharpness in generation tasks. Therefore, we use the $L_1$ distance in our method. The expression to minimize the $L_1$ distance is

$$\min_G L_{L1}(G) = E_{x,x' \sim p_{data}(x,x')}[\| x - x' \|_1]. \tag{2.3}$$

The objective function for such conditional GAN network is the sum of Equations (2.2) and (2.3).

### 2.2 Image Synthesis

**View Transformations:** Existing works on viewpoint transformation have been conducted to synthesize novel views of the same objects [19, 81, 106]. Zhou et al. [106] proposed models that learn to copy the pixel information from input view and utilize them to preserve the identity and
structure of the objects to generate new views. Tatarchenko et al. [81] trained an encode-decoder network to obtain 3D representation models of cars and chairs which they later used to generate different views of an unseen car or chair image. Dosovitskiy et al. [19] learned generative models by training on 3D renderings of cars, chairs and tables and synthesized intermediate views and objects by interpolating between views and models.

In Chapters 3 and 4, we propose generative adversarial network based architectures to learn view transformations between the aerial and the ground viewpoints.

**Geometry-guided Synthesis:** Song et al. [78] propose geometry-guided adversarial networks to synthesize identity-preserving facial expressions. The facial geometry is used as a controlled input to guide the network to synthesize facial images with desired expressions. Similar work by [38] improves the visual quality of synthesized images by enforcing a mechanism to control the shapes of the objects. They map the generator’s output to a mean shape and implicitly enforce the geometry of the objects and also add skip connections to transfer priors to the generated objects.

In Chapter 4, we conduct homography transformation of aerial image to ground perspective to guide the cross-view image synthesis task.

**Image Inpainting:** Pathak et al. [61] generated missing parts of images using networks trained jointly with adversarial and reconstruction losses and produced sharp and coherent images. Yeh et al. [96] tackle the problem of image inpainting by searching for the encoding of the corrupted image that is closest to another image in the latent space and passing it through the generator to reconstruct the image. The closeness is defined based on the weighted context loss of the corrupted image, and a prior loss that penalizes unrealistic images. Yang et al. [95] propose a multi-scale patch synthesis approach for high-resolution image inpainting by jointly optimizing on image content and texture constraint.
In Chapter 4, we inpaint the missing regions of the homography transformed image to synthesize realistic images.

2.3 Image Matching

Image geo-localization has been tackled as an image matching task [2, 25, 98, 101] in computer vision community. Early works on image based geo-localization [70, 86, 100] employed the handcrafted features for matching the query and the gallery images from the same (ground) view. Hays et al. [24] proposed a data-driven approach to estimate the distribution over geographical location from a single image. Researchers followed up with cross-view image geo-localization [44, 77, 34, 71, 3] and matched the features between the aerial and ground images. The hand-crafted features for geo-localization include SIFT descriptors[100], Bag of words [77], VLAD descriptors [34], and building facades [3].


In chapter 5, we use the synthesized images to bridge the domain gap between the ground and the aerial images. We propose approach to fuse the synthesized image features with the query image features to learn robust feature representation for the query image.
2.4 Video Localization and Trajectory Smoothing

With success of image localization methods, their extensions to video localization has been recently explored. However, a limited amount of research on video geo-localization has been reported. Authors in [88, 22] estimate trajectories of streetview videos by comparing the SIFT [49] features of query and gallery frames. Recent work by Heng et al. [26] explores cross-view matching for autonomous vehicle navigation in street-view. Works by Hu and Lee [31] predict the trajectory of a moving ground vehicle by matching the street-view panorama to aerial images with a strong assumption that the initial pose (position and heading) of the moving vehicle is known. Earlier works by [6, 53] perform street view navigation of agents by streetview-to-streetview matching using reinforcement learning. The scope of these works is limited to learning image based features, and they do not explore the joint feature learning by exploiting the temporal proximity between the frames in the query video.

Earlier work by Hakeem et al. [22] discarded the noisy outliers from the GPS predictions and used the remaining GPS points as control points for a b-spline to interpolate the remaining (discarded) locations to smooth the trajectories. Chazal et al. [8] propose data-driven trajectory smoothing framework by moving the noisy GPS points to the barycenter of their nearest neighbors in feature space. Authors in [88] smooth the noisy trajectories by using a Minimum Spanning Trees (MST) based trajectory reconstruction algorithm and eliminate trajectory loops or noisy estimations. Recent work by Hu et al. [31] utilize visual odometry readings of the vehicle in Particle Filter algorithm [83] to smooth the initial predictions of the vehicle location.

In chapter 6, we explore transformer-based framework to learn temporally coherent features for the query video frames as well as geographically smooth features by the application of the proposed GPS loss. Additionally, we propose deep-learning based architecture for trajectory smoothing, as discussed in chapter 6.
2.5 Geo-localization Datasets

The existing geo-localization datasets can be broadly grouped into two categories: same-view matching and cross-view image datasets. Earlier works by [99, 24] build street-view image datasets to conduct the same-view image matching. Some popular datasets for image-based localization with ground and satellite pairs include CVUSA[102], CVACT[47], Vo and Hays [90], UCF-OP [67]. Tian et al. [84] collect cityscale streetview and bird’s eye view image pairs; whereas Zheng et al. [104] collect image from three platforms: synthetic drones, satellites and ground cameras.

The video dataset by Majdik et al. [51] was collected by flying a camera-mounted micro aerial vehicle (MAV) recording the scene from 10-20 meters above the ground and capturing the frontal view of the buildings. They build a reference set of images from Google Street-View data, however, their dataset is limited to a 2 km trajectory in downtown Zurich, Switzerland. Yu et al. [97] collect large scale driving dataset, BDD, covering four different regions of the USA; New York, San Francisco, Berkeley and Bay Area.

In chapter 3, we utilized the geolocalization dataset for cross-view image synthesis task as there was no benchmark dataset for the problem. For this we used CVUSA, and Vo and Hays dataset partially (from Dayton region). For experiments on chapter 4, we used SVA dataset [58], a synthetic dataset collected from game engine. For image matching work in chapter 5, we conducted experiments on benchmark CVUSA dataset and also collected a new dataset from Orlando and Pittsburgh regions of the USA. And finally, for the work in chapter 6, we constructed a large scale video geolocalization dataset by utilizing the BDD dataset as query and collecting corresponding gallery images from Google Street View images.
The inception of Transformer network [89] based on attention mechanisms for long term sequence modelling to solve the language translation task has gained a lot of popularity and has been widely used in different applications. [18] proposed BERT (Bidirectional Encoder Representations from Transformers) that pre-trained deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. Fu et al. [20] propose dual attention network with position and channel attention modules for scene segmentation task. Different applications of attention network include text to speech synthesis [41], text summarization [72], object localization [11], audio-visual event localization [93], video action localization [10, 62].

In chapter 6, we propose deep neural network to learn long-term sequence modelling of video frame features. We also proposed transformer-based trajectory smoothing network to determine noisy GPS locations in the estimated trajectory and regress the offsets to smooth the noisy estimates.

Features at different layers of deep neural networks are essentially the multi-resolution features of the same image. Abundance of literature has explored features at multiple scales [29, 55, 68, 48, 43] for applications like key-point detection, human pose estimation, semantic segmentation. FPN (Feature Pyramid Network) [46], HyperNet [37], ION [4] explored multi-scale features for object detection. Earlier, Hypercolumns [23] were created from multi-layer features and used for object segmentation and localization. Building upon this work, we also aggregate the features at multiple scales to efficiently obtain robust representation of the images in chapter 5.
2.7 Summary

In this chapter, we discussed several previous works that are related to our work in the dissertation. Earlier works in image synthesis conduct image transformations for small changes in viewpoints between source and target images. They didn’t attempt to synthesize images between large transformations that exist between aerial and ground images; which we did in our first two works. In the chapter 3, we used GANs to learn cross-view transformations and in the chapter 4, we utilized geometrical cues (i.e. homography) between the two views to facilitate the synthesis task. Then, in the chapter 5 on cross-view image matching, earlier approaches learnt the view-invariant features between the ground and aerial images for image matching but didn’t attempt to learn transformations between the views before matching. We are the first to utilize the cross-view synthesized images in cross view image matching problem. In Chapter 6, on video geo-localization work, earlier methods learnt image level features for video frames by treating the frames independently and didn’t explore to utilize the temporal dimension in learning features for frames in the video. We are the first to learn the features for the clip by exploiting the long term temporal dependencies in the frames using Transformer network.
CHAPTER 3: CROSS VIEW IMAGE SYNTHESIS USING CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS

The work in this Chapter has been published in the following paper:


In this chapter, we explore the relationships between the aerial and ground view images by solving the task of cross-view image synthesis. Cross-view image synthesis is a task of synthesizing an aerial image as an output that represents the same scene as the input ground-view image; and vice versa. We do so by using Generative Adversarial Network (GAN) based encoder-decoder architectures. The GAN network is conditioned on the input image to generate its transformed image from the cross-view.

The remainder of the chapter is organized as follows: Section 3.1 describes the proposed frameworks in detail, how semantic segmentation is used as a regularizer to guide the cross-view image synthesis; Section 3.2 deals with the datasets and the experimental setup used for the experiments in this work; Section 3.3 presents the detailed analysis of the qualitative and quantitative results on two datasets used in this work; and Section 3.4 summarizes this chapter.

3.1 Methodology

We propose two novel architectures based on conditional generative adversarial network (cGAN) for solving the task of cross-view image synthesis. We learn to generate the target view image
as well as target view segmentation map and show that by penalizing the network on quality of synthesized target view segmentation map will help the full network to improve the synthesis of the target view image.

3.1.1 Crossview Fork (X-Fork) Network

The block diagram of our first architecture, known as Crossview Fork (X-Fork), is shown in Figure 3.1. The generator network is forked to synthesize images as well as semantic segmentation maps. The inherent idea behind this architecture is multi-task learning by the generator network. When the generator is enforced to learn the semantic class of the pixels together with the image synthesis, this helps to improve the image synthesis task.

The generator network of the X-Fork architecture is shown in Figure 3.2. The first six blocks of decoder share the weights. This is because the image and segmentation map contain a lot of shared features. The final two blocks are forked (have different weights) to synthesize the real image and the semantic segmentation map. The number of kernels used in each layer (block) of the generator are shown below the blocks. The discriminator architecture is taken from the baseline [33].
architectural details are explained in Subsection 3.2.2.

Considering the synthesis of the ground level imagery \(I_g\) from aerial image \(I_a\), the input image \(I_a\) is the conditional input to the network. Following Equation 2.2, the conditional GAN loss is expressed as shown in Equation (3.1).

\[
\min_G \max_D L_{\text{cGAN}}(G, D) = E_{I_g, I_a \sim p_{\text{data}}(I_g, I_a)}[\log D(I_g, I_a)] + E_{I_a, I_g' \sim p_{\text{data}}(I_a, I_g')}[\log(1 - D(I_g', I_a))],
\]

(3.1)

where, \(I_g' = G(I_a)\).

Similarly, following Equation 2.3, the \(L_1\) loss on images is represented by Equation (3.2),
\[
\min_G L_{L1}(G; I_g, I'_g) = E_{I_g, I'_g \sim p_{data}(I_g, I'_g)}[\| I_g - I'_g \|_1],
\]

(3.2)

Since the semantic segmentation maps are synthesized by our proposed network, we also include the \(L_1\) distance between the ground-truth segmentation and the generated segmentation map into the loss, as shown in Equation 3.3.

\[
\min_G L_{L1}(S_g, S'_g) = E_{S_g, S'_g \sim p_{data}(S_g, S'_g)}[\| S_g - S'_g \|_1],
\]

(3.3)

Thus, the final loss function for the X-Fork network will be the sum of adversarial loss, \(L_1\) loss on images and \(L_1\) loss on segmentation maps, as shown in Equation (3.4).

\[
L_{X-Fork} = L_{cGAN}(G, D) + \lambda L_{L1}(G; I_g, I'_g) + \lambda L_{L1}(G; S_g, S'_g)
\]

(3.4)

Here, \(\lambda\) is the balancing factor for the losses.

For ground to aerial synthesis, \(I_g\) is the conditional input to the network and the roles of \(I_a\) and \(I_g\) are reversed in the Equations (3.5) and (3.6).

\[
\min_G \max_D L_{cGAN}(G, D) = E_{I_g, I_a \sim p_{data}(I_g, I_a)}[\log D_1(I_g, I_a)] + E_{I_g, I'_a \sim p_{data}(I_g, I'_a)}[\log (1 - D_1(I'_a, I_g))],
\]

(3.5)

where, \(I'_a = G(I_g)\).

Similarly, following Equation 2.3, the \(L_1\) loss on images is represented by Equation (3.6).
\[
\min_{G_1} L_{L1}(G_1; I_a, I'_a) = E_{I_a,I'_a \sim p_{data}(I_a,I'_a)}[||I_a - I'_a||_1],
\]

(3.6)

### 3.1.2 Crossview Sequential (X-Seq) Network

Our second architecture uses a sequence of two cGAN networks as shown in Figure 3.3. The first network generates cross-view images. This is similar to the baseline network that generates images only. The second network receives images from the first generator as conditioning input to synthesize the segmentation map in the same view. Thus, the first network is a cross-view cGAN while the second one is an image-to-segmentation cGAN. The whole architecture is trained end-to-end so that both cGANs learn simultaneously. Intuitively, the input-output dependency between the cGANs constrains the generated images and the segmentation maps, and in effect improves the quality of the generated outputs. Training the first network to generate better cross-view images enhances generation of better segmentation maps by the second generator. At the same time, the feedback from the better trained second network forces the first network to improve its generation. Thus, when both networks are trained in tandem, better quality images are generated compared to the baseline.

Replacing \( G \) and \( D \) in Equations (3.5) and (3.6) by \( G_1 \) and \( D_1 \), respectively, we obtain the equivalent expressions for losses of cross-view cGAN network in this architecture. For the image-to-segmentation cGAN network, the images generated by \( G_1 \) are considered as conditioning inputs.

We now express the cGAN loss for this network as

\[
\min_{G_2} \max_{D_2} L_{cGAN}(G_2, D_2) = E_{S_g \sim p_{data}(S_g)}[\log D_2(S_g, I'_g)] + E_{S'_g \sim p_{data}(S'_g)}[\log(1 - D_2(S'_g, I'_g))],
\]

(3.7)
where, \( I'_g = G_1(I_a) \) and \( S'_g = G_2(I'_g) \). The \( L_1 \) loss for the image-to-segmentation network is

\[
\min_{G_2} L_{L1}(G_2; S_g, S'_g) = E_{S_g, S'_g \sim p_{data}}[|| S_g - S'_g ||_1],
\]

(3.8)

The overall objective function for the X-Seq network is

\[
L_{X-Seq} = L_{cGAN}(G_1, D_1) + \lambda L_{L1}(G_1) + L_{cGAN}(G_2, D_2) + \lambda L_{L1}(G_2).
\]

(3.9)

Equation (3.9) is optimized during the training to learn the parameters \( G_1, D_1, G_2 \) and \( D_2 \).
3.2 Experimental Settings

3.2.1 Dataset

For the experiments in this work, we use the cross-view image dataset provided by Vo et al. [91]. This dataset consists of more than one million pairs of street-view and overhead view images collected from 11 different cities in the US. We select 76,048 image pairs from Dayton and create a train/test split of 55,000/21,048 pairs. We call it Dayton Dataset. The images in the original dataset have resolution of $354 \times 354$. We resize them to $256 \times 256$.

We also recruit the CVUSA Dataset [92] for direct comparison of our work with Zhai et al. [102]. This dataset consists of 35,532/8,884 train/test split of image pairs. Following Zhai et al., the aerial images are center-cropped to $224 \times 224$ and then resized to $256 \times 256$. We only generate a single camera-angle image rather than the panorama. To do so, we take the first quarter of the ground level images and segmentations from the dataset and resize them to $256 \times 256$ in our experiments. Please see Figure 3.9 for some images from the CVUSA dataset.

The two networks, X-Fork and X-Seq, learn to generate the target view images and segmentation maps conditioned on source view image. Training procedure requires the images as well as their semantic segmentation maps. The CVUSA dataset has annotated segmentation maps for ground view images, but for Dayton dataset such information is not available. To compensate, we use one of the leading semantic segmentation methods, known as the RefineNet [43]. This network is pre-trained on outdoor scenes of the Cityscapes dataset [13] and is used to generate the segmentation maps that are utilized as ground truth (pseudo-label) maps. These semantic maps have pixel labels from 20 classes (e.g., road, sidewalk, building, vegetation, sky, void, etc). Figure 3.4 shows image pairs from the dataset and their segmentation masks overlaid in both views. As can be seen, the segmentation mask (label) generation process is far from perfect since it is unable to segment parts
3.2.2 Network and Implementation Details

We use the conditional GAN architecture of [33] as the baseline and call it Pix2pix. The generator is an encoder-decoder network with blocks of Convolution, Batch Normalization [32] and activation layers. We present details of encoder and decoder used for higher resolution (256×256) image generation in Table 3.1. The table shows the operations involved in each layer as well the size of output after each block. Leaky ReLU with a slope of 0.2 is used as the activation function in the encoder, whereas the decoder has ReLU activation except for its final layer where Tanh is used. The first three blocks of the decoder have a Dropout layer in between Batch normalization and activation layer, with dropout rate of 50%.

Figure 3.4: Original ground and aerial image pairs (first row) from the training set and the images with segmentation masks from pre-trained RefineNet [43] overlaid on original images (second row).
Table 3.1: Network Details of encoder and decoder used in GANs. Different operations carried out at each block are shown along with output dimensions. Conv: Convolution, lReLU: Leaky ReLU, BN: Batch Normalization, ReLU: Rectified Linear Unit, UpConv: Upconvolution.

<table>
<thead>
<tr>
<th>G Block</th>
<th>Operations</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL₁</td>
<td>Conv, lReLU</td>
<td>64×128×128</td>
</tr>
<tr>
<td>e CBL₂</td>
<td>Conv, BN, lReLU</td>
<td>128×64×64</td>
</tr>
<tr>
<td>n CBL₃</td>
<td>Conv, BN, lReLU</td>
<td>256×32×32</td>
</tr>
<tr>
<td>c CBL₄</td>
<td>Conv, BN, lReLU</td>
<td>512×16×16</td>
</tr>
<tr>
<td>o CBL₅</td>
<td>Conv, BN, lReLU</td>
<td>512×8×8</td>
</tr>
<tr>
<td>d CBL₆</td>
<td>Conv, BN, lReLU</td>
<td>512×4×4</td>
</tr>
<tr>
<td>e CBL₇</td>
<td>Conv, BN, lReLU</td>
<td>512×2×2</td>
</tr>
<tr>
<td>r CBL₈</td>
<td>Conv, BN, lReLU</td>
<td>512×1×1</td>
</tr>
<tr>
<td>UBDR₁</td>
<td>UpConv, BN, Dropout, ReLU</td>
<td>512×2×2</td>
</tr>
<tr>
<td>d UBDR₂</td>
<td>UpConv, BN, Dropout, ReLU</td>
<td>512×4×4</td>
</tr>
<tr>
<td>e UBDR₃</td>
<td>UpConv, BN, Dropout, ReLU</td>
<td>512×8×8</td>
</tr>
<tr>
<td>c UBR₄</td>
<td>UpConv, BN, ReLU</td>
<td>512×16×16</td>
</tr>
<tr>
<td>o UBR₅</td>
<td>UpConv, BN, ReLU</td>
<td>256×32×32</td>
</tr>
<tr>
<td>d UBR₆</td>
<td>UpConv, BN, ReLU</td>
<td>128×64×64</td>
</tr>
<tr>
<td>e UBR₇</td>
<td>UpConv, BN, ReLU</td>
<td>64×128×128</td>
</tr>
<tr>
<td>r UT₈</td>
<td>UpConv, Tanh</td>
<td>3×256×256</td>
</tr>
</tbody>
</table>

The generator of X-Fork method has a single encoder and two symmetrical decoders; one for image synthesis and the other for segmentation map synthesis. The decoders share weights except for the final two layers. For X-Seq architecture, we have a sequence of two GANs; the generators are encoder-decoder networks same as the baseline.

The discriminators used in all three methods have common architecture. It is similar to the encoder part of generator except that the final two convolution operations have a stride of 1 and the final activation layer is Sigmoid instead of Tanh. The discriminator output is a prediction of the probability of its input being real.

For lower resolution (64×64) experiments, CBL₇ and CBL₈ blocks are removed from encoder, and UBDR₁ and UBDR₂ are dropped from decoder. Since the input resolution is reduced by 4, the
second and third values in dimensions of ‘output size’ for each block in the table also get reduced by four.

The used convolutional kernels are 4×4 with a stride of 2. The upconvolution in the decoder is Torch[12] implementation of SpatialFullConvolution, and upsamples the input by 2. For the encoder and the discriminator, convolutional operation downsamples the images by 2. No pooling operation is used in the networks. The $\lambda$ used in Equations (3.4) and (3.9) is the balancing factor between the GAN loss and $L_1$ loss. Its value is fixed at 100. Following the idea to smooth the labels by [80] and demonstration of its effectiveness by Salimans et al. [69], we use one-sided label smoothing to stabilize the training process, replacing 1 by 0.9 for real labels. During the training, we utilized different data augmentation methods like random jitter and horizontal flipping of images. The network is trained end-to-end with weights initialized with a random Gaussian distribution with zero mean and 0.02 standard deviation. It is implemented in Torch [12]. The code is publicly available ¹.

### 3.3 Results

Our experiments are conducted in a2g (aerial-to-ground) and g2a (ground-to-aerial) directions on Dayton dataset and a2g direction only on CVUSA dataset. We consider image resolutions of 64×64 and 256×256 on Dayton dataset while for experiments on CVUSA dataset, 256×256 resolution images are used.

First, we run experiments on lower resolution images (64×64) for proof of concept. Encouraging qualitative and quantitative results in this resolution motivated us to apply our methods to higher resolution (256×256) images. The lower resolution experiments are carried out for 100 epochs

¹https://github.com/kregmi/cross-view-image-synthesis
with batch size of 16, whereas the higher resolution experiments are conducted for 35 epochs with batch size of 4.

We conduct experiments on CVUSA dataset for comparison with Zhai et al.’s work [102]. Following their setup, we train our architectures for 30 epochs, using the Adam optimizer and moment parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

It is not straightforward to evaluate the quality of synthesized images [5]. In fact, evaluation of GAN methods continues to be an open problem [82]. A common evaluation method is to show the generated images to human observers and ask their opinion about the images. Human judgment is based on the response to the question: Is this image (image-pair) real or fake? Alternatively, the images generated by different generative models can be pitted against each other and the observer is asked to select the image that looks more real. But in experiments involving natural scenes, such evaluation methods are more challenging as multiple factors often affect the quality of the generated images. For example, the observer may not be sure whether to base his judgment on better visual quality, higher sharpness at object boundaries, or more semantic information present in the image (e.g., multiple objects in the images, more details on objects, etc). Therefore, instead of behavioral experiments, we illustrate qualitative results in Figures 3.5, 3.6 and 3.9 and conduct an in-depth quantitative evaluation on test images of two datasets.

### 3.3.1 Qualitative Evaluation

For 64×64 resolution experiments, the networks are modified by removing the last two blocks of CBL from discriminator and encoder, and the first two blocks of UBDR from decoder of the generator. We run experiments on all three methods. Qualitative results are depicted in Figure 3.5. The results affirm that the networks have learned to transfer the image representations across the views. Generated ground level images clearly show details about road, trees, sky, clouds, and
pedestrian lanes. Trees, grass, road, house roofs are well rendered in the synthesized aerial images.

![Diagram of a2g and g2a synthesis methods](image)

Figure 3.5: Example images generated by different methods in lower (64 × 64) resolution in a2g (aerial-to-ground) and g2a (ground-to-aerial) directions.

For 256×256 resolution synthesis, we conduct experiments on all three architectures and illustrate the qualitative results on Dayton dataset in Figures 3.6, 3.7, 3.8 and the results on CVUSA dataset in Figures 3.9 and 3.10. For Dayton dataset, we observe that the images generated in higher resolution contain more details of objects in both views and are less granulated than those in lower resolution. Houses, trees, pedestrian lanes, and roads look more natural. Test results on CVUSA dataset show that images generated by proposed methods are visually better compared to Zhai et al. [102] and Pix2pix [33] methods.

Figure 3.7 displays the qualitative results on Dayton dataset for aerial to ground (a2g) synthesis.
Figure 3.6: Example images generated by different methods in higher (256 × 256) resolution in a2g (aerial-to-ground) and g2a (ground-to-aerial) directions.

We observe that, addition of $L_1$ loss between real and generated semantic segmentation images while training our networks helps to generate images that are semantically related to ground truth compared to Pix2pix[33]. Images in rows 1, 3, 4, 5, 6 and 8 show that the proposed methods are successful at generating roads at correct locations. Houses are generated with structural details in images of rows 1, 5, 7 and 8. Sidewalks are well represented in generated images of rows 7 and 8. X-Fork learns to synthesize cars in images of rows 1 and 3. X-Seq tries to generate car in rows 1, 7 and 8.

The ground to aerial (g2a) generation on Dayton dataset is shown in Figure 3.8. The street-view image has house on right side of road and the X-Seq method is able to generate the house even
Figure 3.7: Qualitative results for Aerial to Ground level (a2g) synthesis using different methods on Dayton dataset. The networks learn to generate road, sidewalk (rows 2, 5, 7, 8), trees, houses (rows 1, 3, 5, 7, 8), pole (rows 2, 3, 4, 5) in the images.
Figure 3.8: Qualitative results for Ground to Aerial (g2a) synthesis using different methods on Dayton dataset.
though it is not present in ground truth image. But the numerous cars present in the true aerial image do not have correspondence in street-view image. So, they are not generated by the methods. The networks have learned to generate shadows as seen around trees in second row images. The ground truth aerial image in second row does not show the house that is visible in conditioning street-view image. So, the houses are present in images generated by all three methods. Images in rows 2, 5 and 7 show the X-Fork network has learned to generate cars.

The generated images in aerial view look more distorted primarily because the generated roads are not parallel at edges. This is due to occlusions of road by trees in the aerial images. The major challenge in g2a synthesis is that networks need to estimate the scene beyond the small field of view of ground level image. The generated images should preserve the homogeneity of structures in the scene. This causes greater discrepancy between real and generated images in outer regions of images.

The qualitative results for a2g generation on CVUSA dataset are visualized in Figures 3.9 and 3.10. Our proposed methods generate much better images than Zhai et al. [102] and Pix2pix [33]. The proposed networks are successful at capturing the semantic information in aerial images and transforming them to the target view.

3.3.2 Quantitative Evaluation

The quantitative results of our experiments on both datasets are presented in Tables 3.2-3.5. 64 × 64 and 256 × 256 in column headers of the tables refer to results obtained for two resolutions of Dayton dataset. Next, we discuss the quantitative measures used to evaluate our methods.

Inception Score: A common quantitative GAN evaluation measure is the Inception Score [69]. The core idea behind the inception score is to assess how diverse the generated samples are within
Figure 3.9: Qualitative results of our methods and baselines on CVUSA dataset in a2g direction. First two columns show true image pairs, next four columns show images generated by Zhai et al. [102], Pix2pix[33], X-Fork and X-Seq methods, respectively.

a class while being meaningfully representative of the class at the same time.

\[
\text{Inception Score (IS)} = \exp(E_x D_{KL}(p(y|x)||p(y))) \tag{3.10}
\]

where, \(x\) is a generated sample and \(y\) is its predicted label.

We can not use the Inception model trained on ImageNet [16] because the datasets that we use
Figure 3.10: Qualitative results for Aerial to Ground level (a2g) synthesis using different methods on CVUSA dataset.
include natural outdoor images that do not fit into ImageNet classes. To solve this, we use the AlexNet model [39] trained on Places dataset [105] with 365 categories to compute the inception score. The Places dataset has images similar to those in our datasets. The scores are reported in Table 3.2. The scores for X-Fork generated images are closest to that of real data distribution for Dayton dataset in lower resolution in both directions and also in higher resolution in a2g direction. The X-Seq method works best for CVUSA dataset and for g2a synthesis in higher resolution over Dayton dataset.

We observe that the confidence scores predicted by the pre-trained model on our dataset are dispersed between classes for many samples and not all the categories are represented by the images. Therefore, we compute inception scores on Top-1 and Top-5 classes, where "Top-k" means that top k predictions for each image are unchanged while the remaining predictions are smoothed by an epsilon equal to \(1 - \sum(\text{top-k predictions})/(n-k \text{ classes})\). Results on top-k classes follow a similar pattern as in all classes (except for Top-1 class on lower resolution in g2a over Dayton dataset).
In addition to inception score, we compute the top-k prediction accuracy between real and generated images. We use the same pre-trained Alexnet model to obtain annotations for real images and class predictions for generated images. We compute top-1 and top-5 accuracies. Results are shown in Table 3.3. For each setting, accuracies are computed in two ways: 1) considering all images, and 2) considering real images whose top-1 (highest) prediction is greater than 0.5. Below each accuracy heading, the first column considers all images whereas the second column computes accuracies the second way. For lower resolution images on Dayton dataset and for experiments on CVUSA dataset, X-Fork method outperforms the remaining methods. For higher resolution images, our methods show dramatic improvements over Pix2pix in the \textit{a2g} direction, whereas X-Seq works best in the \textit{g2a} direction.

**Table 3.3: Accuracies: Top-1 and Top-5.**

<table>
<thead>
<tr>
<th>Dir. Methods</th>
<th>64×64</th>
<th>256×256</th>
<th>CVUSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top-1</td>
<td>Top-5</td>
<td>Top-1</td>
</tr>
<tr>
<td>Zhai et al. [102]</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Pix2pix [33]</td>
<td>7.90</td>
<td>15.33</td>
<td>27.61</td>
</tr>
<tr>
<td>X-Fork</td>
<td>16.63</td>
<td>34.73</td>
<td>46.35</td>
</tr>
<tr>
<td>X-Seq</td>
<td>4.83</td>
<td>5.56</td>
<td>19.55</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pix2pix [33]</td>
<td>1.65</td>
<td>2.24</td>
<td>7.49</td>
</tr>
<tr>
<td>g2a X-Fork</td>
<td>4.00</td>
<td>16.41</td>
<td>15.42</td>
</tr>
<tr>
<td>X-Seq</td>
<td>1.55</td>
<td>2.99</td>
<td>6.27</td>
</tr>
</tbody>
</table>

**KL(model | data):** We next compute the KL divergence between the model generated images and the real data distribution for quantitative analysis of our work, similar to some generative works [9, 56]. We again use the same pre-trained Alexnet as in the previous subsection. The lower KL score implies that the generated samples are closer to the real data distribution. The scores are provided in Table 3.4. As it can be seen, our proposed methods generate much better results than existing generative methods on both datasets. X-Fork generates images very similar to
Table 3.4: KL Divergence between model and data distributions.

<table>
<thead>
<tr>
<th>Dir.</th>
<th>Method</th>
<th>64 × 64</th>
<th>256 × 256</th>
<th>CVUSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>a2g</td>
<td>Zhai et al. [102]</td>
<td>–</td>
<td>–</td>
<td>27.43 ± 1.63</td>
</tr>
<tr>
<td></td>
<td>Pix2pix [33]</td>
<td>6.29 ± 0.8</td>
<td>38.26 ± 1.88</td>
<td>59.81 ± 2.12</td>
</tr>
<tr>
<td></td>
<td>X-Fork</td>
<td><strong>3.42 ± 0.72</strong></td>
<td>6.00 ± 1.28</td>
<td><strong>11.71 ± 1.55</strong></td>
</tr>
<tr>
<td></td>
<td>X-Seq</td>
<td>6.22 ± 0.87</td>
<td><strong>5.93 ± 1.32</strong></td>
<td>15.52 ± 1.73</td>
</tr>
</tbody>
</table>

Real distribution in all experiments except on the higher resolution a2g experiment where X-Seq is slightly better than X-Fork.

**SSIM, PSNR and Sharpness Difference:** As in some generative works [52, 73, 59], we also employ Structural-Similarity (SSIM), Peak Signal-to-Noise Ratio (PSNR) and Sharpness Difference measures to evaluate our methods.

SSIM measures the similarity between the images based on their luminance, contrast and structural aspects. SSIM value ranges between -1 and +1. A higher value means greater similarity between the images being compared. It is computed as

$$SSIM(I_g, I'_g) = \frac{(2\mu_g\mu'_{I_g} + c_1)(2\sigma_g\sigma'_{I_g} + c_2)}{(\mu^2_g + \mu'^2_{I_g} + c_1)(\sigma^2_g + \sigma'^2_{I_g} + c_2)}$$

(3.11)

PSNR measures the peak signal-to-noise ratio between two images to assess the quality of a transformed (generated) image compared to its original version. The higher the PSNR, the better is the quality of generated image. It is computed as

$$PSNR(I_g, I'_g) = 10\log_{10}\left(\frac{\text{max}_{I_g}^2}{\text{mse}}\right)$$

(3.12)
where, \( \text{mse}(I_g, I'_g) = \frac{1}{n} \sum_{i=1}^{n} (I_g[i] - I'_g[i])^2 \),

and \( \text{max}_{I'_g} = 255 \) (maximum pixel intensity value).

Sharpness difference (SD) measures the loss of sharpness during image generation. To compute the sharpness difference between the generated image and the true image, we follow [52] and compute the difference of gradients between the images as

\[
SD(I_g, I'_g) = 10 \log_{10} \left( \frac{\text{max}_{I'_g}^2}{\text{grads}} \right),
\]

(3.13)

where, \( \text{grads} = \frac{1}{N} \sum_i \sum_j (|\nabla_i I_g + \nabla_j I_g| - |\nabla_i I'_g + \nabla_j I'_g|) \)

and, \( \nabla_i I = |I_{i,j} - I_{i-1,j}| \), \( \nabla_j I = |I_{i,j} - I_{i,j-1}| \).

Sharpness difference in Equation (3.13) is inverse of \( \text{grads} \). We would like the \( \text{grads} \) to be small, so the higher the overall score the better.

Table 3.5: SSIM, PSNR and Sharpness Difference(SD) between real data and samples generated using different methods.

<table>
<thead>
<tr>
<th>Dir. Methods</th>
<th>64×64</th>
<th>256×256</th>
<th>CVUSA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SSIM</td>
<td>PSNR</td>
<td>SD</td>
</tr>
<tr>
<td>Zhai et al. [102]</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Pix2pix [33]</td>
<td>0.4808</td>
<td>19.4919</td>
<td>16.4489</td>
</tr>
<tr>
<td>a2g X-Fork</td>
<td>0.4921</td>
<td>19.6273</td>
<td>16.4928</td>
</tr>
<tr>
<td>X-Seq</td>
<td>0.5171</td>
<td>20.1049</td>
<td>16.6836</td>
</tr>
<tr>
<td>Pix2pix [33]</td>
<td>0.3675</td>
<td>20.5135</td>
<td>14.7813</td>
</tr>
<tr>
<td>g2a X-Fork</td>
<td>0.3682</td>
<td>20.6933</td>
<td>14.7984</td>
</tr>
<tr>
<td>X-Seq</td>
<td>0.3663</td>
<td>20.4239</td>
<td>14.7657</td>
</tr>
</tbody>
</table>

The scores are reported in Table 3.5. Over Dayton dataset, X-Seq model works the best in \text{a2g} direction while X-Fork outperforms the rest in the \text{g2a} direction. On CVUSA, X-Fork improves
Table 3.6: Frechet Inception Distance (FID) scores on Dayton and CVUSA datasets.

<table>
<thead>
<tr>
<th>Dir. Method</th>
<th>Dayton(64×64)</th>
<th>Dayton(256×256)</th>
<th>CVUSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>[102]</td>
<td>-</td>
<td>-</td>
<td>571.32</td>
</tr>
<tr>
<td>a2g X-Pix2pix</td>
<td>361.10</td>
<td>64.97</td>
<td>644.45</td>
</tr>
<tr>
<td>X-Fork</td>
<td>227.16</td>
<td>71.47</td>
<td>185.42</td>
</tr>
<tr>
<td>X-Seq</td>
<td>676.14</td>
<td><strong>46.34</strong></td>
<td><strong>89.12</strong></td>
</tr>
<tr>
<td>g2a X-Pix2pix</td>
<td>323.54</td>
<td>71.35</td>
<td>-</td>
</tr>
<tr>
<td>X-Fork</td>
<td><strong>239.94</strong></td>
<td>80.03</td>
<td>-</td>
</tr>
<tr>
<td>X-Seq</td>
<td>388.85</td>
<td><strong>69.73</strong></td>
<td>-</td>
</tr>
</tbody>
</table>

over Zhai et al. by 5.03% in SSIM, 8.93% in PSNR, and 12.35% in Sharpness difference.

**FID Score:** An alternative metric to evaluate the quality of generated images is by computing Frechet Inception Distance [28] between the generated samples and the real images. We use the same AlexNet model (as above) pretrained on the Places Dataset to compute the FID score. The lower value of FID score for a method, the better. The FID scores that we obtain in this work are relatively larger than numbers reported in other works mainly because of the variations in the image statistics of the Places Dataset used during the training and our test images.

The FID scores are presented in Table 3.6. On Dayton dataset, X-Fork performs the best on lower resolution images while X-Seq works the best on higher resolution images in both directions a2g and g2a. X-Seq works the best on the CVUSA dataset closely followed by X-Fork network.

Because there is no consensus in evaluation of GANs, we had to use several scores. Theis et al. [82] show that these scores often do not agree with each other and this was observed in our evaluations as well. So, it is difficult to infer whether X-Fork or X-Seq is better. We find that the proposed methods are consistently superior to the baselines.
3.3.3 Generated Segmentation Maps

Figure 3.11: Segmentation overlay: Original image pairs are shown in first column, second column shows segmentation maps obtained from RefineNet [43] overlaid on original images; these segmentation maps are used as ground truth segmentation maps. The next two columns show segmentation maps from our proposed X-Fork and X-Seq methods overlaid on original images. The color representation of common semantic classes from Dayton dataset are provided below the images.

Our methods generate semantic segmentation maps along with the real images in cross-view. In Figure 3.11, we show the qualitative comparison of segmentation maps synthesized by our methods compared to ground truth segmentation maps.

Figure 3.11 shows the pairs of real images in first column followed by the segmentation maps
overlaid on real images in columns two, three and four. The second column shows ground truth segmentation maps, the third and fourth columns show the segmentation maps synthesized by our X-Fork and X-Seq methods respectively. The ground-truth (pseudo-label) segmentation maps obtained by using state-of-the-art RefineNet [43] pretrained on Cityscapes dataset [13] overlaid on real image pairs are shown in second column. These semantic maps have pixel labels from 20 classes (‘road’, ‘sidewalk’, ‘building’, ‘wall’, ‘fence’, ‘pole’, ‘trafficlight’, ‘trafficsign’, ‘vegetation’, ‘terrain’, ‘sky’, ‘person’, ‘rider’, ‘car’, ‘truck’, ‘bus’, ‘train’, ‘motorcycle’, ‘bicycle’, ‘void’). Most common classes in Dayton dataset are: road, sidewalk, building, vegetation, terrain, sky, etc. The segmentation maps for aerial images show missed classification (‘void’ class) for road, buildings, cars in images of row 2, 3 and 6. This is because RefineNet was primarily trained on street-view images and is tested on aerial images here. Similarly, the segmentation maps generated by proposed methods are overlaid on image pairs and illustrated in third and fourth columns respectively. The overlaid images show that the network is able to learn the semantic representations of object classes.

On ground level segmentations, our methods perform well to capture sky, road, terrain and vegetation. This is best represented in rows 1 and 4. Images in row 6 show trees misclassified as sky at top right corner by X-Seq method and is better classified by X-Fork. The aerial image segmentation has some misclassification. The images in row 1 show road is classified as void by X-Fork, as in its ground truth segmentation from RefineNet, whereas it is classified as vegetation by X-Seq. Images in row 3, 4 and 6 show road and buildings are missed in ground truth segmentation. The roads are partially segmented for images of row 3 and the methods do pretty good for row 4. The problem still exists in generated segmentations of row 6.

The generation of aerial segmentation is more challenging because all object classes may not be visible in street-view images. This normally happens when we have many objects in aerial view images that are occluded in street-view. It is the same reason why our methods work well in images
of row 4 but not for images of row 6.

Table 3.7: Evaluation Scores (per-class accuracy and mean Intersection over union (mIOU)) for segmentation maps generated by X-Fork and X-Seq methods for aerial-to-ground (a2g) and ground-to-aerial (g2a) synthesis. The scores

<table>
<thead>
<tr>
<th>Methods</th>
<th>a2g</th>
<th>g2a</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Per-class accuracy</td>
<td>mIOU</td>
</tr>
<tr>
<td>X-Fork</td>
<td>0.6262</td>
<td>0.4163</td>
</tr>
<tr>
<td>X-Seq</td>
<td>0.4783</td>
<td>0.3187</td>
</tr>
</tbody>
</table>

For quantitative analysis, we report the per-class accuracy and meanIOU scores for the synthesized segmentation maps. Per-class accuracy is measured as the proportion of correctly labeled pixels for each class and is then averaged over the classes. Mean intersection-over-union (mIOU) is computed as the intersection over union for each semantic class and is then averaged over the classes. Intersection over union is computed as the ratio of true positives to the sum of true positives, false positives and false negatives.

The segmentation maps generated by our methods are compared against the segmentation maps obtained by applying RefineNet [43] to the target images (ground truths). We compute per-class accuracies and mean IOU for the most common classes in our datasets: ‘vegetation’, ‘road’ and ‘building’ in aerial segmentation maps, plus the ‘sky’ in ground segmentations. The scores are reported in Table 3.7. The scores affirm that our cross-view synthesis networks are successful at transforming the semantics between the two views. Even though X-Fork does better than X-Seq, we find that both methods achieve good scores for segmentation.
3.3.4 Visualization of Quantitative Evaluation (Scatterplot)

We conduct further quantitative analysis of our methods on synthesized ground level images by visualizing the scatter plots of SSIM scores of 21048 test images of Dayton dataset in Figure 3.12. Each dot in the scatter plot represents a test image; its SSIM score for baseline (Pix2pix) and proposed methods are obtained by projecting the point to the x-axis and y-axis respectively.

The left scatter plot shows the comparison of SSIM scores between X-Fork and Pix2pix methods and the right scatter plot shows the comparison between X-Seq and Pix2pix methods. The scatter plots show the superiority of our proposed methods over the baseline, since most of the plots are above the diagonal (line: $y = x$). We also show the images at few x points in the scatter plots. The generated images look similar to the ground truth even for points where SSIM values are low.

Figure 3.12: Scatterplot of SSIM scores obtained using different methods (left: X-Fork vs Pix2pix and right: X-Seq vs Pix2pix) for test images of Dayton dataset. Images for different points in the scatterplot are shown.
Figure 3.13: The first two columns show aerial and ground image pairs, third column shows corresponding street-view image synthesized by X-Seq method and next three columns show nearest images in the training set retrieved by computing $L_1$ distance between generated ground image and training set images.

3.3.5 $kNN$

Here, we test whether our proposed architectures have actually learned to transform the representations between images in two views rather than just memorizing the blocks from training images to generate new ones. We visualize the real image pairs and the synthesized ground image followed by three images from the training set that are closest to the synthesized images in terms of $L_1$ distance in Figure 3.13. As shown, the generated images have subtle differences with the training set images implying that our network has indeed learned important semantic representations in input view needed to transform the source image to target view.
3.4 Summary

In this Chapter, we introduced two conditional generative adversarial networks, X-Fork and X-Seq; and synthesized the images between aerial and ground views. We demonstrated that generating the semantic segmentation maps together with the images in target view helps the networks to learn to synthesize better images compared to the baseline method where only target view images are synthesized. We conducted extensive qualitative and quantitative evaluations to validate the effectiveness of our methods. Using higher resolution (256 × 256) images provided significant improvements in visual quality and added more details to synthesized images. In the next chapter, we discuss cross-view image mage synthesis by leveraging the geometrical cues between the views to guide the synthesis.
CHAPTER 4: GEOMETRY-GUIDED CROSS VIEW IMAGE SYNTHESIS

The work in this chapter has been published in the following paper:


In chapter 3, we presented cross-view image synthesis architectures (X-Fork and X-Seq) based on conditional Generative Adversarial Networks. The drawback of such architectures is that they are purely end-to-end learning based methods and do not incorporate geometrical cues in the training. The networks need to learn the geometrical constraints pertinent between the ground and aerial views. In this chapter, we extend the work presented in Chapter 3 by introducing geometry-guided learning to ease the task of cross-view image synthesis. Since there exists an overlapping field of view (FOV) between aerial and ground images, this shared region can be geometrically transformed from aerial to the ground view and can be preserved to assist the cross-view synthesis that we explore in this chapter. Here, we use homography to transform the aerial images to the ground viewpoint as a preprocessing step and use the transformed image as input to the network to guide the cross-view synthesis.

The remainder of the chapter is organized as follows: Section 4.1 describes the proposed framework in detail, how homography transformed image is obtained for the aerial images together with the steps in synthesizing the cross-view images; Section 4.2 discusses the datasets and implementation details; Section 4.3 provides the results and experimental evaluations; and finally, Section 4.4 summarizes this chapter.
4.1 Methodology

In this chapter, we are motivated to exploit the geometrical relationships that exists between between the aerial and ground images and we attempt to preserve them to synthesize meaningful cross-view images. Our hypothesis is that the majority of the scene from the first view (aerial) can be transformed into the second perspective (ground level) using the homography and this should ease the synthesis task.

In this section, we first discuss how homography matrix is computed to get homography transformed image from aerial input image, followed by the discussion about the baseline methods and the proposed architectures for the task of cross-view image synthesis. We also illustrate how our proposed method "H-Regions" is able to preserve the source pixels as well as fill in the missing regions for cross-view image synthesis.

4.1.1 Homography Transformation

We use homography as a preprocessing step to transform the visual features from aerial images to the ground perspective. We compute homography matrix between the ground and aerial images using point correspondences between the images. Given \( \mathbf{x} \) and \( \mathbf{x}' \) contain a set of corresponding points in two planes, the homography equation is represented as shown in Equation 4.1.

\[
\mathbf{x}' = \mathbf{Hx},
\]

(4.1)

where \( \mathbf{H} \) is homography transformation. Equation 4.1 can be further expressed as shown in Equation 4.2.
Given a set of four 2D to 2D point correspondences between the images in two views, the homography matrix \( H \) is determined by solving the Equation 4.2 by minimizing the least square error.

We started by computing the SIFT features between the aerial and ground image pairs and then finding keypoints in two images. We used the keypoints to determine the homography matrix and then transform the images between the views. This method could not find the corresponding points in the images in two views, most likely because of very large perspective variation between the views, and thus was not successful in obtaining plausible homography matrix for transforming the images between the views.

Thus, we manually determine the corresponding points in the two views. First, we randomly pick a pair of images from the dataset. We then manually select four points in the aerial image and find their corresponding locations in the ground-view image. We use these points to compute the homography matrix \( H \) that transforms the aerial image to ground view and vice versa. We then use the computed homography matrix to transform all the aerial images in the dataset to ground perspective. Surprisingly, thus obtained homography matrix works well for all the image pairs in the dataset and avoids expensive computations for each pair of images separately. This is possible because the locations of aerial and ground view cameras are fixed in the SVA dataset used in this chapter.

We utilize the homography transformed images in two ways: a) use them as input to the end-to-
end learning approaches discussed in chapter 3, and b) by preserving the transformed pixels in the
target view and inpaint the missing regions in the target view images.

We next discuss the baseline approaches followed by the proposed methods.

4.1.2 Baselines

The naive way to approach this task of cross-view image synthesis is to consider it as an image to
image translation problem. This is done by feeding in the input images to a deep neural network
and obtaining the transformed image as the output. We run the experiments in the following
settings as our baselines.

Cross-view Image-to-Image Translation (X-Pix2pix): For this, the generator is an encoder-
decoder network that takes in an image in first view (eg. aerial) as input and learns to generate the
image in the other view (eg. ground) as output.

Cross-view Image-to-Image Translation with Stacked Outputs (X-SO): The network takes
an image in first view as input and generates an output image of 6 channels, the first 3 channels
correspond to the RGB image and the next three channels represent the segmentation map. The L1
loss and adversarial loss are computed over six channels of output and the corresponding ground
truth images.

Crossview Fork (X-Fork): The X-Fork architecture, as shown in Figure 3.1 and explained in
Section 3.1.1 consists of a generator that is forked to synthesize two outputs of 3 channels each,
the first output is the RGB image and the second output is the segmentation map both in target
view. The inherent idea behind this architecture is multi-task learning by the generator network.
When the generator is enforced to learn the semantic class of the pixels together with the image
synthesis, this helps to improve the image synthesis task. The generated segmentation map serves
as an auxiliary output.

**Crossview Sequential (X-Seq):** Our second architecture, X-Seq uses a sequence of two cGAN networks as shown in Figure 3.3, and explained in Section 3.1.2. The first network performs cross-view image synthesis and the generated image is fed to the second network as a conditioning input to synthesize its corresponding segmentation map. This two-stage end-to-end learning of image and segmentation map synthesis produces improved image quality compared to the network without the second stage.

**Cross-view Pix2pix with Homography (H-Pix2pix):** This approach uses the same baseline Pix2pix network with the difference that the input image is the homography transformed image instead of its real aerial image.

**Cross-view Stacked Outputs with Homography (H-SO):** The network takes the homography transformed image as input and generates target view image and the segmentation map stacked together as a 6-channel output; first 3 channels for image and the next three channels for its segmentation map.

4.1.3 Proposed approach

The approach we present in this section utilizes the homography transformed images as input to the network. Once the image from the first view is transformed into the second perspective using the homography, these pixels in the overlapping field of view can be preserved during cross-view synthesis. The missing regions in the transformed images are mostly related to sky and buildings.

The hypothesis behind this idea is that the use of transformed aerial images as input should ease the synthesis of ground level images compared to synthesizing by feeding the aerial images directly. There are three variations of the proposed approach discussed below:
Cross-view Fork with Homography (H-Fork): Here, we use the homography transformed image $I_{ah}$ as input to the X-Fork network architecture proposed in subsection 3.1.1.

Crossview Sequence with Homography (H-Seq): In this setup, we feed the homography transformed image $I_{ah}$ rather than the original input image $I_a$ as input to the X-Seq network of subsection 3.1.2.

Crossview Regions with Homography (H-Regions): In this method, we attempt to preserve the structural details visible in the aerial view images and guide the network to transfer those details to the synthesized ground view images. For this, we use the homography transformed image as input to our method and solve the synthesis task in following subtasks:

Subtask 1: The homography transformed image ($I_{ah}$ in Figure 4.1) has a large portion of missing region ($R_1$) in the image. Our first task is to fill in the missing region in the transformed images. We use an encoder-decoder network that takes $I_{ah}$ as input and generates only the upper half of the image ($I_{g'}$).

![Figure 4.1](image)

Figure 4.1: An aerial image $I_a$ shown in (a) is first transformed to street-view perspective using homography ($I_{ah}$), shown in (b). The transformed image, $I_{ah}$ needs inpainting in upper box region ($R_1$) and further processing in car region ($R_2$), as shown in (c). These two regions are generated and the region in between is copied from homography transformed image $I_{ah}$ to obtain $I_{g'}$. We further train $I_{g'}$ to smooth the region boundaries and add realism to it. $I_g$ is the corresponding ground truth image (e).
Subtask II: The street-view images are recorded by a camera mounted on a car’s dashboard. Therefore, all the street-view images contain a part of the car’s hood around the lower central region ($R_2$) in them which can be seen in image $I_g$ in Figure 4.1. Note that, this scenario is present in SVA dataset and can be avoided in real applications where the dash camera can be mounted such that the hood of the car is not captured in the frames. The homography transformed image $I_{ah}$ also has a car around that region which has been transformed from the aerial view of the car but it does not realistically represent a car in street-view. To address this, we mask a probable car-region and train a small network dedicated to learn mapping of the car region. This helps generate a realistic car region in ground view images (See region $R_2$ in image $I_g^\prime$).

Once we have images generated from the first two tasks, we copy them to their respective spatial locations in homography transformed image $I_{ah}$ creating a street-view image $I_{g^\prime}$ and preserving the pixels of remaining regions. Copying pixels in this manner helps us preserve the structural information that has been transformed using homography. The problem with this approach is that images do not look realistic at region boundaries. So, we further train another network to add realism to this image.

Subtask III: GANs are successful at generating images that look very realistic to human eyes. Here, we train a conditional GAN architecture on $I_g^\prime$. For this, we first define bands around the region boundaries as shown in Figure 4.1. We formulate the loss function to preserve (by copying) the pixel information outside the bands to the output image while at the same time adding realism to the whole image. This step helps a lot to improve the visual quality of the synthesized image.

We now define our loss functions for the subtasks. For subtask one, we use a conditional GAN network to inpaint missing regions in $I_{ah}$ by optimizing the network on adversarial and $L_1$ losses for the missing regions only. For subtask two, we only consider region $R_2$ by masking out the remaining regions in input and output images and optimizing for adversarial and $L_1$ losses for
car region only. Once we have results from the above two subtasks, we compute $I'_g$ as shown in Equation 4.3.

\[
I'_g = I_{Inpaint} \odot M_1 + I_{car} \odot M_2 + I_{ah} \odot (M - M_1 - M_2) \tag{4.3}
\]

Here, $I_{Inpaint}$ is the image generated from the inpainting network of subtask one. $I_{car}$ is the car image generated in subtask two. $M_1$ and $M_2$ are 3-channel binary masks for regions $R_1$ and $R_2$, $M$ being the 3-channel all ones image. The masks $M_1$ and $M_2$ are manually computed looking at the homography transformed image ($I_{ah}$) in Figure 4.1. This was done for a single frame only and worked well for all the images in the dataset. If the hood of the car was not visible in the street-view image, we wouldn’t even need region $R_2$ and correspondingly mask $M_2$. $I'_g$ is fed to the realism network to generate the final image in the target view. $\odot$ is the element-wise product.

4.2 Experimental Settings

4.2.1 Dataset

In this section, we present details about the dataset used in this work.

**SVA Dataset:** The Surround Vehicle Awareness (SVA) dataset [58] is a synthetic dataset collected from Grand Theft Auto V (GTAV) video game. The game camera is toggled between frontal and bird’s eye view to simultaneously capture images in the two views at each game time step. We use the train/test split as provided in the dataset. The original dataset has 100 sets of training set images and 50 sets of test set images. The consecutive frames in each set are very similar to each other, so we use every tenth frame to remove redundancy in the dataset. Finally, we have a training set of 46,030 image pairs and a test set of 22,254 image pairs. The images are resized to $256 \times$
256 for experiments in this work. Sample images from SVA dataset are shown in the leftmost and rightmost columns of Figure 4.2. We use this dataset for experiments in aerial-to-ground (a2g) direction only.

The proposed Fork and Sequence networks learn to generate the target view images and segmentation maps conditioned on source view image or their homography transformed image. Training procedure requires the images as well as their semantic segmentation maps. The SVA dataset doesn’t provide the annotated segmentation maps for the images. To compensate, we use one of the leading semantic segmentation methods, known as the RefineNet [43]. This network is pre-trained on outdoor scenes of the Cityscapes dataset [13] and is used to generate the segmentation maps that are utilized as ground truth maps. These semantic maps have pixel labels from 20 classes (e.g., road, sidewalk, building, vegetation, sky, void, etc). Thus obtained segmentation maps are used as ground truth maps for training the proposed methods.

4.2.2 Implementation Details

For our experiments with homography transformed images as inputs, we obtain the homography transformed images from the aerial images as explained in Section 4.1.1.

The conditional GAN architecture used for the experiments in this chapter are the same as explained in Chapter 3. The generator is an encoder-decoder network with blocks of Convolution, Batch Normalization [32] and activation layers. Leaky ReLU with a slope of 0.2 is used as the activation function in the encoder, whereas the decoder has ReLU activation except for its final layer where Tanh is used. The first three blocks of the decoder have a Dropout layer in between Batch normalization and activation layer, with dropout rate of 50%. The discriminator is taken as it is from the [33]. A minor modification was done on generator architecture. We removed two blocks of CBL and UBDL from the generator architecture, primarily to save training time. We observed
that removal of these blocks did not have much impact on the quality of synthesized images. The $\lambda_1$ and $\lambda_2$ used in the objective function for different networks are the balancing factors between the GAN loss and the $L_1$ loss. For realism task in the H-Regions method, $\lambda_1 = 5$ for adversarial loss and $\lambda_2 = 2$ for pixel-wise loss worked the best.

Also, note that the synthesized semantic maps are 3-channel RGB images which effectively mitigated the class imbalance among the semantic classes. This was primarily done to consider all 20 semantic classes during the training and reduce bias towards dominant classes like houses, trees, road and sky. Had the semantic classes been limited to the dominant ones only, it would regularize the synthesized images to not learn the less prevalent objects in the target view images. Also, we brought in some confidence by the success of the pix2pix in synthesizing 3-channel segmentation maps from RGB images.

We train the baseline and the proposed networks for 20 epochs on SVA dataset. For H-Regions, we conduct experiments as follows. For subtask I, we train the network for 20 epochs. For subtask II, we train another network for one epoch only since the network needs to learn the mapping of the car and to preserve its color from source view to the target view. Eventually, we train a final realism network for 5 epochs (subtask III).

4.3 Results

We have conducted experiments in aerial-to-ground (a2g) direction on SVA datasets. We consider the image resolution of $256 \times 256$ for aerial and ground images.

We follow the similar evaluation metrics as used in Chapter 3 for evaluation of synthesized images. We report Inception Score (IS), Accuracy, KL(model||data), SSIM, PSNR, Sharpness Difference and FID Score as quantitative metrics of evaluation. Refer Section 3.3.2 for more details on each
of the evaluation metrics. Additionally, we also run a user study to conduct a subjective evaluation of synthesized images.

### 4.3.1 Qualitative Evaluation

We conduct the homography experiments on SVA dataset and report the qualitative evaluation. This is because the aerial view image for SVA dataset contains high overlap with the field of view of street-view image and thus application of homography to preserve the details from aerial image seemed valid for this dataset, compared to Dayton and CVUSA datasets used in chapter 3.

The qualitative results on the SVA dataset for aerial to street-view synthesis is shown in Figure 4.2. The proposed methods are capable at generating roads, car-hood, markers on road, sky and other details in the images. We observe that the images generated by the proposed H-Regions method contain more details around the central regions. This is due to enforcing the network to preserve those details from the aerial images.

We also conduct the user study to evaluate the quality of synthesized images as explained next.

**User Study:** For the subjective evaluation of different methods, we run a user study on the images synthesized using these methods. We show an aerial image along with the corresponding images in ground view synthesized using seven different methods to the users/subjects. We specifically ask each user to select the most realistic ground image that also contains the most visual details from the aerial view image.

We conducted the study over 100 test images on 10 subjects to compare the images synthesized by different methods on the SVA dataset. The results are presented in Table 4.1. The most preferred method is H-Regions, closely contested by H-Seq and X-Seq methods. The results illustrate the following: a) The use of homography transformed input drastically outperforms corresponding ex-
<table>
<thead>
<tr>
<th>Aerial</th>
<th>Homography</th>
<th>X-Pix2pix</th>
<th>X-SO</th>
<th>X-Fork</th>
<th>X-Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Image" /></td>
<td><img src="image2.jpg" alt="Image" /></td>
<td><img src="image3.jpg" alt="Image" /></td>
<td><img src="image4.jpg" alt="Image" /></td>
<td><img src="image5.jpg" alt="Image" /></td>
<td><img src="image6.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image7.jpg" alt="Image" /></td>
<td><img src="image8.jpg" alt="Image" /></td>
<td><img src="image9.jpg" alt="Image" /></td>
<td><img src="image10.jpg" alt="Image" /></td>
<td><img src="image11.jpg" alt="Image" /></td>
<td><img src="image12.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image13.jpg" alt="Image" /></td>
<td><img src="image14.jpg" alt="Image" /></td>
<td><img src="image15.jpg" alt="Image" /></td>
<td><img src="image16.jpg" alt="Image" /></td>
<td><img src="image17.jpg" alt="Image" /></td>
<td><img src="image18.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image19.jpg" alt="Image" /></td>
<td><img src="image20.jpg" alt="Image" /></td>
<td><img src="image21.jpg" alt="Image" /></td>
<td><img src="image22.jpg" alt="Image" /></td>
<td><img src="image23.jpg" alt="Image" /></td>
<td><img src="image24.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image25.jpg" alt="Image" /></td>
<td><img src="image26.jpg" alt="Image" /></td>
<td><img src="image27.jpg" alt="Image" /></td>
<td><img src="image28.jpg" alt="Image" /></td>
<td><img src="image29.jpg" alt="Image" /></td>
<td><img src="image30.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image31.jpg" alt="Image" /></td>
<td><img src="image32.jpg" alt="Image" /></td>
<td><img src="image33.jpg" alt="Image" /></td>
<td><img src="image34.jpg" alt="Image" /></td>
<td><img src="image35.jpg" alt="Image" /></td>
<td><img src="image36.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image37.jpg" alt="Image" /></td>
<td><img src="image38.jpg" alt="Image" /></td>
<td><img src="image39.jpg" alt="Image" /></td>
<td><img src="image40.jpg" alt="Image" /></td>
<td><img src="image41.jpg" alt="Image" /></td>
<td><img src="image42.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image43.jpg" alt="Image" /></td>
<td><img src="image44.jpg" alt="Image" /></td>
<td><img src="image45.jpg" alt="Image" /></td>
<td><img src="image46.jpg" alt="Image" /></td>
<td><img src="image47.jpg" alt="Image" /></td>
<td><img src="image48.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 4.2: Example images generated by different methods in aerial-to-ground (a2g) direction for SVA dataset. Upper panel: First column shows the aerial input images, followed by their homography transformed images in second column. Column 3-6 show images synthesized by different methods using the aerial images directly as input, i.e. without homography transformation. Lower panel: Column 1-5 shows images synthesized by different methods by utilizing homography transformed images as input. Finally, the last column shows the ground truth street-view images.
Table 4.1: % of user preferences over images synthesized by different methods (over 100 images from the SVA dataset).

<table>
<thead>
<tr>
<th></th>
<th>X-Pix2pix</th>
<th>X-Fork</th>
<th>X-Seq</th>
<th>H-Pix2pix</th>
<th>H-Fork</th>
<th>H-Seq</th>
<th>H-Regions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>7.8</td>
<td>16.8</td>
<td>12.6</td>
<td>14.2</td>
<td>18.2</td>
<td>26.4</td>
</tr>
</tbody>
</table>

Experiments with untransformed aerial image as input, and b) Users preferred the images synthesized using H-Regions because of the method’s ability to preserve the pixel information onto the target view.

4.3.2 Quantitative Comparison

We discuss the quantitative evaluation on SVA dataset next. The quantitative results of our experiments on SVA dataset is illustrated in Table 4.2.

**Inception Score:** H-Regions generates images that have the inception score closest to that of real data.

**Accuracy:** H-Seq method performs best in terms of accuracy.

**KL(model || data):** Images synthesized using X-Seq method have the closest distribution to the ground truth distribution among all the methods.

**FID Score:** The H-Regions performs the best in terms of FID score in SVA dataset. The H-models perform better than their X-counterparts.

**SSIM, PSNR, and SD:** X-Seq achieves the highest numbers in terms of SSIM and PSNR whereas H-Regions has the best SD. Also, H-Pix2pix already performs very good compared to X-Pix2pix because homography simplified the learning task by transforming the image to the target view. H-
Table 4.2: Quantitative evaluation of samples generated using different methods on SVA Dataset in a2g direction. IS - Inception Score, Acc. - Accuracy

<table>
<thead>
<tr>
<th>Methods</th>
<th>X-Pix2pix</th>
<th>X-SO</th>
<th>X-Fork</th>
<th>X-Seq</th>
<th>H-Pix2pix</th>
<th>H-SO</th>
<th>H-Fork</th>
<th>H-Seq</th>
<th>H-Regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>*IS, all ↑</td>
<td>2.0131</td>
<td>2.4951</td>
<td>2.1888</td>
<td>2.2232</td>
<td>2.1906</td>
<td>2.3202</td>
<td>2.3202</td>
<td>2.2394</td>
<td>2.6328</td>
</tr>
<tr>
<td>*IS, Top-1 ↑</td>
<td>1.7221</td>
<td>1.8940</td>
<td>1.9776</td>
<td>1.9842</td>
<td>1.9507</td>
<td>1.9410</td>
<td>1.9525</td>
<td>1.9892</td>
<td>2.0732</td>
</tr>
<tr>
<td>*IS, Top-5 ↑</td>
<td>2.2370</td>
<td>2.6634</td>
<td>2.3664</td>
<td>2.4344</td>
<td>2.4069</td>
<td>2.7340</td>
<td>2.3918</td>
<td>2.4385</td>
<td>2.8347</td>
</tr>
</tbody>
</table>

| Acc. (Top-1, all) ↑ | 8.5961    | 7.5146| 17.3794| 19.5056| 18.0706    | 5.2444| 18.0182| 20.7391| 15.4803   |
| Acc. (Top-1, 0.5) ↑ | 30.3288   | 30.9507| 53.4725| 57.1010| 54.8068    | 26.4697| 51.0756| 57.5378| 48.0767   |
| Acc. (Top-5, 0.5) ↑ | 29.9102   | 38.9822| 63.5045| 65.3005| 62.3072    | 31.9527| 62.8166| 67.4649| 56.8994   |

| SSIM ↑            | 0.3206    | 0.4552| 0.4235 | 0.4638| 0.4327    | 0.4457| 0.424   | 0.4249| 0.4044    |
| SD ↑              | 17.0254   | 17.5285| 16.9371| 17.4138| 16.9468    | 17.3876| 16.8653| 17.5616| 17.6858   |

| FID Score ↓       | 859.66    | 443.79| 129.16 | 118.70| 117.13    | 1452.88| 109.43  | 95.12 | 88.78     |

*Inception Score for real (ground truth) data is 3.0347, 2.3886 and 3.3446 for all, top-1 and top-5 setups respectively.

methods outperform their X- counterparts for most of the evaluation metrics.

Because there is no consensus in evaluation of GANs, we had to use several scores. [82] show that these scores often do not agree with each other and this was observed in our evaluations as well. Nonetheless, we find that the proposed methods are consistently superior to the baselines in terms of quantitative and qualitative evaluations.

4.4 Summary

We explored image generation using conditional GANs between two drastically different views by exploiting the geometrical cues between the views. Using homography to guide the cross-view synthesis allowed preserving the overlapping regions between the views. We conducted extensive qualitative and quantitative evaluations and validate the effectiveness of our proposed methods.
In the next chapter, we utilize X-Fork network to synthesize cross-view images and use them in cross-view image matching pipeline to solve the image geo-localization problem.
CHAPTER 5: CROSS VIEW IMAGE MATCHING

The work in this Chapter has been published in the following paper:


In this chapter, we further explore the relationships between aerial and ground level images by solving the task of cross-view image matching. Cross-view image matching is a task of retrieving an aerial image from a gallery or reference database corresponding to the query ground image, which represents the same scene as a query ground level image; and vice versa. Given the geo-tags of the reference images are known, the image retrieval can subsequently be applied to solve the task of geo-localization by assigning the geo-tag of the retrieved reference image to a query image. For cross-view matching, we learn the view-independent feature representations corresponding to the images from aerial and ground views. Building upon the work presented in the previous chapters for synthesis of the cross-view images, we employ the synthesized images in the process of learning the query and the gallery features. In this work, the ground level panorama images (with 360 degree FOV) are used to represent the street-view scene, instead of frontal view images (with 90 degree FOV) employed previously.

The remainder of the chapter is organized as follows: Section 5.1 describes the proposed framework in detail, how synthesized images are employed together with the real images in feature learning process for the query and gallery images; Section 5.2 deals with the datasets and implementation details used for the experiments; Section 5.3 provides the results and the experimental evaluations and ablation studies; and Section 5.4 summarizes the work in this chapter.
Figure 5.1: Joint Feature Learning: Inputs to this network are $I_g$ and $I_a$ and outputs are $f_g$ and $f_a$. Employing auxiliary loss between $f_a'$ and $f_a$ helps to pull features $f_g$ and $f_a$ closer, and minimize the domain gap between two features than when training two-stream network on $(I_g, I_a)$ pairs. The branch in the middle (dotted box filled with cyan) is used during the training only.

5.1 Methodology

We propose a novel method to bridge the domain gap between street-view and aerial images by leveraging the synthesized aerial images using GANs. We learn the representations of synthesized aerial images jointly with ground and aerial image representations. Additionally, we fuse the complementary representations of ground images with the representations of their corresponding synthesized aerial images to learn robust query representations of ground images. Also, we exploit the edgemaps of input images to provide GANs with the notion of object shapes and boundaries and facilitate the cross-view image synthesis.

Joint Feature Learning: Along with the query ground panorama $I_g$ and gallery aerial images $I_a$, our work utilizes the synthesized aerial image $I_{a'}$ during the feature learning (training). We propose to learn the feature representations for image triads: query ground panorama, $I_g$, synthesized aerial image, $I_{a'}$ from ground panorama and aerial image $I_a$ jointly, so that the synthesized aerial image
representations $f_{a'}$ pushes the image representations $f_g$ and $f_a$ closer to each other.

The joint feature learning architecture is shown in Figure 5.1. The encoder blocks are shown in green (for ground image) and blue (for aerial images) triangles. Each encoder consists of deep convolutional architecture as described in subsection 5.2.2. We elegantly exploit the inherent multiscale pyramidal structure of features at multiple layers of deep neural networks. We consider the features from the final three convolutional layers, conv_6, conv_7 and conv_8 layers. These features are aggregated and followed by a fully connected layer to obtain the feature representation for images in each view.

The encoders for aerial and street-view images do not share the weights. Since the cross-view images are captured from different viewpoints, the visual entities exhibit drastic domain changes. The two encoders operate on these sets of diverse images, so it is understandable that the weight sharing is not a good choice. On the other hand, the encoders for $I_{a'}$ and $I_a$ share the weights, since both images represent the aerial domain. This way, the aerial encoders learn weights suitable for the synthesized image $I_{a'}$ as well as the real image $I_a$. Thus, $f_{a'}$ effectively forces the features $f_a$ to be closer to $f_g$ and bridges the domain gap between the two views. This is possible because the transformed image $I_{a'}$ captures representations of $I_g$ which are easier for the network to learn from $I_{a'}$ than it would be when learning directly from $I_g$.

This strategy leverages the synthesized images at training time, but does not require them during the testing. The auxiliary loss between $I_{a'}$ and $I_a$ influences the aerial image encoder to learn representations for aerial images by considering the synthesized aerial image. We train our network jointly on these image triads ($I_g$, $I_{a'}$ and $I_a$) using weighted soft-margin ranking loss [30], which is explained next.

*Weighted Soft-margin Triplet Loss:* Consider a feature embedding $f_g$ of ground-level image, $f_{a-\text{pos}}$ of the corresponding matching aerial image and a non-matching aerial image feature $f_{a-\text{neg}}$. The
triplet loss [27] aims to bring the matching feature \( f_{a\_pos} \) closer to \( f_g \) while at the same time pushes \( f_{a\_neg} \) away. Here, if \( d_p \) is the Euclidean distance between positive samples \((f_g, f_{a\_pos})\) and \( d_n \) is the Euclidean distance between negative/non-matching samples \((f_g, f_{a\_neg})\), we try to minimize \( d_p \) as well as maximize \( d_n \). The triplet loss is expressed as shown below:

\[
L_{triplet} = \max(0, m + d_p - d_n),
\]

(5.1)

where, \( m \) is a margin that specifies a minimum distance between non-matching pairs.

In order to avoid the necessity of explicitly deciding the margin for triplet loss, soft-margin triplet loss is popular and is expressed as given in Equation 5.2 below:

\[
L_{soft} = \ln(1 + e^d),
\]

(5.2)

where \( d = d_p - d_n \).

In our work, we use the weighted soft margin triplet loss [30] as given in Equation 5.3:

\[
L_{weighted} = \ln(1 + e^{\alpha d}).
\]

(5.3)

We use \( \alpha = 10 \), which results in better convergence than \( \alpha = 1 \).

We incorporate the auxiliary loss between the synthesized aerial images, \( I_a' \), and the real aerial images, \( I_a \), along with the loss between the real aerial, \( I_a \), and the ground images, \( I_g \), for joint feature learning using the Equation 5.4 below:

\[
L_{joint} = \lambda_1 L_{weighted}(I_g, I_a) + \lambda_2 L_{weighted}(I_a', I_a).
\]

(5.4)
Figure 5.2: Feature Fusion. Inputs to this network are $I_g$ and $I_a$ and outputs are $f_{g^*}$ and $f_{a^*}$. $f_g$ and $f_{a'}$ are concatenated and passed through fully-connected layer (FC) to get their fused representation $f_{g^*}$.

Here, $\lambda_1$ and $\lambda_2$ are balancing factors between the losses for $(I_g, I_a)$ and $(I_{a'}, I_a)$ pairs respectively.

**Feature Fusion:** In the previous method, the synthesized aerial image is used during the training only, for bridging the domain gap between the real aerial and ground view images; but is neglected during testing. Since the features of the synthesized image contain complementary information that assist in joint feature learning, we attempt to further exploit them. We fuse the ground image features $f_g$ with synthesized aerial image features $f_{a'}$ and find a robust representation $f_{g^*}$ for the query ground image.

The fusion architecture is shown in Figure 5.2. We use the trained joint feature learning network as feature extractor for our feature fusion task. We first concatenate the features from ground query image with the features from synthesized aerial image. The concatenated features need to be refined to obtain a generalized representation for query image $f_{g^*}$. We achieve this by passing through a fully-connected layer in the upper stream. The features $f_a$ from the lower stream need to be optimized against the refined features from upper fully-connected layer. So, we add a fully-
connected layer in the lower stream that learns the generalized representations, \( f_a \), for the aerial images. During the testing, the fused feature representation \( f_g \) for query image \( I_g \) is compared against the representations \( f_a \) for aerial images for image matching.

5.2 Experimental Setup

This section deals with the datasets we used and the experimental setups we followed in our work.

5.2.1 Datasets

We conduct experiments on CVUSA dataset [102] to compare our work with existing methods. We also collect a new dataset, UCF-OP dataset, from urban areas of Orlando and Pittsburgh with geo-information. The other benchmark dataset, GT-Crossview [91] doesn’t contain the ground level panorama, thus making it infeasible to synthesize meaningful aerial image. Also, the GT-Crossview dataset has aligned image pairs in training set, whereas unaligned image pairs in test set with no direction information, so the synthesized aerial images for test case will be randomly oriented relative to aerial images in the reference database, thus it is not possible to use this dataset in our framework.

**CVUSA:** CVUSA is a benchmark dataset for cross-view image matching with 35,532 satellite and ground-panorama image pairs for training and 8,884 pairs for testing. Aerial images are \( 750 \times 750 \) and ground-panorama are \( 224 \times 1232 \) in resolutions. Sample images from this dataset are shown in Figure 5.3.

**Orlando-Pittsburgh (OP) dataset:** The existing public datasets on cross-view image matching do not provide geo-information. So, the evaluation of matching algorithms in terms of accuracy
in distance (meters) is not feasible. Also, the images on the CVUSA dataset are collected from the rural areas that largely cover land and vegetation as shown in Figures 5.3, 5.6 and 5.8. A new dataset that covers urban areas can help to evaluate the generalization of the proposed methods. To compensate those issues, we collect a new dataset of cross-view image pairs from urban areas of two US cities, Orlando and Pittsburgh. Figure 5.11 shows the example images of this dataset. We can observe that this dataset contains images of mainly urban areas with buildings and roads and less vegetation, contrasting to the CVUSA dataset.
The dataset contains 1,910 training and 722 testing pairs of aerial and ground-panorama images. The resolutions are 640 × 640 for aerial images and 416 × 832 for panoramas. Though small-scale, this dataset will be useful for future research in this direction.

5.2.2 Implementation Details

We present the implementation details of our cross-view synthesis network and the proposed image matching networks in this section.

**Joint Feature Learning network:** Each stream (encoder) of joint feature learning network in Figure 5.1 consists of seven convolutional layers, each followed by ReLU activations. Dropouts are applied after the final three ReLU layers. The features after these dropouts are flattened and then concatenated to obtain multi-scale representation of the input image. This is followed by a fully-connected layer for dimensionality reduction to obtain 1,000-dimensional feature vector for each input. The two-stream baselines are trained from scratch with Xavier initialization. The joint feature learning network is initialized with weights from the two-stream network trained on (I_g, I_a) image pairs and the loss function is optimized as shown in Equation 5.4. We use \( \lambda_1 = 10 \) and \( \lambda_2 = 1 \), weighing more on the loss term for (I_g, I_a) pairs because of their superior performance over (I_{a'}, I_a) in image matching as reported in Table 5.1 and objectively I_{a'} is used as an auxiliary information, only during the training in joint feature learning network.

**Feature Fusion network:** The Feature Fusion network in Figure 5.2 has two fully-connected layers, one each for aerial and ground feature branches. The upper FC layer takes 2000-dimensional fused feature and translates it to a 1000-dimensional feature representation. The input to the lower FC layer is \( f_a \) that is mapped to a 1000 dimensional feature representation. The FC layers are randomly initialized with a uniform distribution with zero mean and 0.005 standard deviation.
The two-stream baselines and the proposed joint feature learning and feature fusion networks are implemented using Tensorflow [1] with Adam optimizer (lr = $10^{-5}$) and dropout = 0.5. A batch size of $B = 30$ for experiments on two-stream networks and $B = 24$ for joint feature learning networks is used. Weighted soft-margin triplet loss is used for training in all the experiments. An exhaustive mini-batch strategy [91] is employed to maximize the number of triplets within each batch. For each image in a batch of $B$ images, we have 1 positive pair and $(B-1)$ negative pairs for each ground image, and $(B-1)$ negative pairs for each aerial image. So, for $B$ images, we have $B$ positive pairs and $2 \times B \times (B-1)$ negative pairs. Further training is continued with in-batch hard negative mining; by training each positive pair against the most negative sample (i.e. smallest distance) in the batch. Code and dataset is publicly available 1.

Cross-View Synthesis network: The generator of cross-view synthesis network, shown as Generator in Figures 5.1 and 5.2 has an encoder and two decoders, similar to the X-Fork architecture in [65]. The input to the encoder is a 4-channel image; 3-RGB channels and an edgemap, stacked together. The decoders generate cross-view image and its segmentation map, for a given input. The network consists of blocks of Convolution, Batch Normalization and Leaky ReLU layers. Convolutional kernels of size $4 \times 4$ with a stride of 2 are used that downsamples the feature maps after each convolution, and to upsample the feature maps after each upconvolution operation. We reshape the features at bottleneck to adjust the feature shape and pass through the decoders. The six blocks of decoders share the weights whereas the final two blocks don’t. The discriminator network has similar architecture to the one used in [65]. We train the GAN end-to-end using Torch[12] implementation. The weights are initialized with a random Gaussian distribution with zero mean and 0.02 standard deviation.

In summary, GAN is first trained to generate the cross-view image $I_a'$ for the ground panorama $I_g$.  

1https://github.com/kregmi/cross-view-image-matching
Next, the synthesized images are used for joint feature learning in our proposed method.

5.3 Results

We present an extensive analysis of our proposed method demonstrating the effectiveness of synthesized images for image retrieval to bridge the domain gap between the cross-view images. We also provide the comparison of our work with the state-of-the-art methods on the CVUSA dataset. Finally, we present an evaluation on geo-localization task on the OP dataset.

5.3.1 Evaluation Metrics

The common metric for evaluation of image based matching task is to compute the recall accuracy. A matching is successful for a query street-view image if the correct match lies within a set of closest images in Euclidean distance of the representative features. We report top-1% accuracy for ease of comparison with previous works. We also report top-1 and top-10 recalls on CVUSA dataset. We also compute the localization accuracy for UCF-OP dataset where the geo-information is available. Here, the localization is considered accurate if the query image is located within a threshold distance in meters from its ground truth position.

5.3.2 Ground-to-Aerial Image Matching: Results on CVUSA Dataset

We evaluate our model variants in terms of retrieval accuracy on the CVUSA dataset [102]. The results are reported in Table 5.1 (first panel).

Baseline Comparison (first and second rows in Table 5.1 (first panel)): The two-stream networks trained employing image pairs \((I_g, I_a)\) and \((I_{a'}, I_a)\), where first image in each tuple is the query,
Table 5.1: Comparison of Top-1, Top-10 and Top-1% recall for the baselines and the proposed approaches (first panel) and with previous methods (second panel) on CVUSA Dataset [102].

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1</th>
<th>Top-10</th>
<th>Top-1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-stream baseline ($I_{a}', I_a$)</td>
<td>10.23%</td>
<td>35.10%</td>
<td>72.58%</td>
</tr>
<tr>
<td>Two-stream baseline ($I_g, I_a$)</td>
<td>18.45%</td>
<td>48.98%</td>
<td>82.94%</td>
</tr>
<tr>
<td>Joint Feat. Learning ($I_w, I_a$)</td>
<td>14.31%</td>
<td>48.75%</td>
<td>86.47%</td>
</tr>
<tr>
<td>Joint Feat. Learning ($I_g, I_a$)</td>
<td>29.75%</td>
<td>66.34%</td>
<td>92.09%</td>
</tr>
<tr>
<td>Feature Fusion</td>
<td>48.75%</td>
<td>81.27%</td>
<td>95.98%</td>
</tr>
<tr>
<td>Workman et al. [92]</td>
<td>-</td>
<td>-</td>
<td>34.3%</td>
</tr>
<tr>
<td>Zhai et al. [102]</td>
<td>-</td>
<td>-</td>
<td>43.2%</td>
</tr>
<tr>
<td>Vo and Hays [91]</td>
<td>-</td>
<td>-</td>
<td>63.7%</td>
</tr>
<tr>
<td>CVM-Net-I [30]</td>
<td>22.53%</td>
<td>63.28%</td>
<td>91.4%</td>
</tr>
<tr>
<td>CVM-Net-II [30]</td>
<td>11.18%</td>
<td>43.51%</td>
<td>87.2%</td>
</tr>
</tbody>
</table>

are the baselines. We observe that the synthesized image $I_{a}'$ as a query performs quite well with 72.58% for top-1% recall but slightly lower than $I_g$ as query (82.94%). This means that the synthesized images capture fair amount of information from the ground panorama whereas they are not yet completely dependable for cross-view image retrieval and we need to consider real ground images as well. This provided us the motivation for joint feature learning.

**Joint Feature Learning** (third and fourth rows in Table 5.1 (first panel)): For joint feature learning, as explained earlier, image triads ($I_g, I_a$ and $I_{a}'$) are used during training and only ($I_g, I_a$) pairs are used during the testing. We report an improvement of about 9% in top-1% retrieval accuracy over two-stream baseline ($I_g, I_a$) by joint feature learning. The improvement suggests that the synthesized aerial images contain features complementary to ground image features that facilitate the network to learn better representations for aerial images during the joint feature learning. The synthesized aerial image as an auxiliary information between the ground and aerial images is successful in forcing them to bring their feature representations closer to each other during the joint feature learning.
Since the representations for $I_g$, $I_a$ and $I_{a'}$ were learned together during joint feature learning, we were curious to evaluate how well the feature representations for $I_{a'}$ perform in image matching. Unsurprisingly, we obtain an improvement of about 14% in top-1% retrieval accuracy over two-stream baseline ($I_{a'}, I_a$). This improvement further consolidated the belief that the learned features for $I_g$ and $I_{a'}$ are complementary to each other and can be fused together to obtain robust descriptor for the ground image.

**Feature Fusion:** (fifth row in Table 5.1 (first panel)): The Feature Fusion approach fuses the synthesized image features with the ground image features to obtain a representative feature for the query. This provides further improvement of 3.89% in top-1% accuracy (compare fourth and fifth rows). The significance of feature fusion can be measured by about 19% improvement in top-1 retrieval accuracy over joint feature learning. This improvement further signifies that the synthesized image features are complementary to street-view image features that should be exploited to obtain better features for cross-view matching. The qualitative results are shown in Figure 5.3. The query ground images and the synthesized aerial images along with five closest images are shown in each row.

**Comparison to Existing Methods:** We compare our work with the previous approaches by [92, 102, 91, 30] on CVUSA dataset [102]. We report the top-1, top-10 and top1-% accuracies for state-of-the-art CVM-Net [30] and our methods. The results are shown in Table 5.1 (second panel). We observe that the Joint Feature Learning outperforms (fourth row in Table 5.1 (first panel)) the previous works and is further boosted by Feature Fusion (fifth row in Table 5.1 (first panel)). We achieve an overall 4.58% improvement over SOTA CVM-Net [30] for top-1% recall accuracy. We obtain significant increments of more than 26% and 18% in top-1 and top-10 accuracies over CVM-Net-I [30]. We also plot top-K recall accuracy for $K = 1$ to 80 for our methods as compared with previous approaches in Figure 5.4. It illustrates that various versions of our proposed method
outperform the existing state-of-the-art approaches for all values of $K$.

**Visualization and Interpretation of Features:** In Figure 5.5, we visualize the aerial and ground image features obtained using the two-stream baseline and the proposed feature fusion methods for 100 images on the CVUSA dataset [102]. The feature representation for each image is a 1000-dimensional vector and we apply t-SNE to learn their two-dimensional embeddings for ease of visualization. The red and cyan circles close to each other or with some overlap represent the features for the ground image and its corresponding ground-truth aerial image respectively in the
subplots.

The scatter-plot for features obtained using the two-stream baseline trained on \((I_g, I_a)\) pairs is shown on the left. We observe that, for each image pair, there is less overlap between the aerial and the ground image features. We also notice that the features from different image pairs are located close to each other, with some instances of red circles overlapping each other.

The scatter-plot for the representations obtained using the proposed feature fusion method trained on image triads \((I_g, I_a, I_{a'} )\) is shown on the right subplot. We observe higher overlap between the features for ground and corresponding aerial image pairs. At the same time, we observe greater separation between the feature embeddings for different image samples.

Thus, the use of synthesized aerial images in our proposed Feature Fusion method are successful in bringing the feature representations of aerial images closer to the representations of ground images and bridging the domain gap between the images from these two drastically different views to improve the matching accuracy.
**Ablation Study:** We conduct the following ablation studies to understand the impact of different choices made in the proposed networks. For the experiments on ablation, the joint feature learning and feature fusion networks are used with specified setups: a) single scale features - only the final layer features are matched, b) global average pooling (GAP) - GAP operation suppresses the spatial dimension of feature maps, substantially reducing the feature size, and c) weight sharing between the encoders for aerial and ground images. All these methods reduce the number of parameters used in the network.

**Single Scale vs. Multi-scale Features:** For this ablation, joint feature learning network with single scale features is trained first followed by experiments using the Feature Fusion network. The features after the final convolutional block (conv_8) are considered as single scale features. These are the representative features for the given input image and are used for matching. We do not employ global average pooling and weight sharing in this ablation for direct comparison of the single-scale vs. multi-scale feature representations. The scores are reported in Table 5.2 (first row for single scale and fourth row for multi-scale features). The results signify that features from conv_6 and conv_7 are also crucial in image matching rather than just using the features from final conv_8 layer only. The results demonstrate the importance of aggregating the multi-scale features for cross-view matching task.

Table 5.2: Ablation Study on CVUSA Dataset [102]. The reported numbers are the retrieval accuracies for Feature Fusion network for specified ablation criteria.

<table>
<thead>
<tr>
<th>Ablation Criteria</th>
<th>Top-1</th>
<th>Top-10</th>
<th>Top-1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Scale Features</td>
<td>8.01 %</td>
<td>32.62 %</td>
<td>74.41%</td>
</tr>
<tr>
<td>Global Avg. Pooling (GAP)</td>
<td>16.13%</td>
<td>51.72%</td>
<td>87.68%</td>
</tr>
<tr>
<td>Weight Sharing</td>
<td>29.94%</td>
<td>68.24%</td>
<td>93.42%</td>
</tr>
<tr>
<td>Multi-scale Features + No GAP + No Wt. Sharing</td>
<td><strong>48.75%</strong></td>
<td><strong>81.27%</strong></td>
<td><strong>95.98%</strong></td>
</tr>
</tbody>
</table>
Pooling vs. No Pooling: We also conduct ablations on using global average pooling [44] in our experiments. Global average pooling is a popular approach to reduce the spatial dimensions of the features and consequently reduce the number of parameters in the network. We experimented with using global average pooling layer before concatenating the features from multiple scales. The result is reported in Table 5.2 (second row for using GAP and fourth row without using GAP, rest of the architecture being the same). We observe that the loss of spatial information in features severely impacts the retrieval performance.

Weight Sharing vs. No Weight Sharing: We believe that the two branches receiving the input images from completely different viewpoints as is the case with aerial and ground-view images should not share the weights. Even though the networks will be looking at same scene contents their representations from the two views are drastically different, thus suggesting that the networks should freely evolve their weights based on the input they receive. The results are reported in Table 5.2 (third row for weight sharing and fourth row for without weight sharing, remainder of the setup being the same). The numbers clearly suggest that no weight sharing is fairly an easy choice, especially looking at the difference of about 18% in Top-1 accuracies.

Failure Examples: We present some failure cases for the proposed Feature Fusion method in Figure 5.6. In each row, We respectively present the query ground image, corresponding synthesized aerial image, image retrieved at Top-1 position, ground-truth aerial image and a number representing the position where the ground-truth aerial image was retrieved.

Row 1 shows that ground truth aerial image consisting of water body in lower right section of the image. The ground image does not provide any information regarding water, so the image matching is challenging. The ground truth is retrieved at 13th position.

In Row 2, we can observe that the top match and ground-truth aerial images are very similar to
<table>
<thead>
<tr>
<th>Ground Query</th>
<th>Synthesized Aerial</th>
<th>Top-1 match</th>
<th>Ground Truth Aerial</th>
<th>Retrieval Index</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Ground Query" /></td>
<td><img src="image2" alt="Synthesized Aerial" /></td>
<td><img src="image3" alt="Top-1 match" /></td>
<td><img src="image4" alt="Ground Truth Aerial" /></td>
<td>13</td>
</tr>
<tr>
<td><img src="image5" alt="Ground Query" /></td>
<td><img src="image6" alt="Synthesized Aerial" /></td>
<td><img src="image7" alt="Top-1 match" /></td>
<td><img src="image8" alt="Ground Truth Aerial" /></td>
<td>37</td>
</tr>
<tr>
<td><img src="image9" alt="Ground Query" /></td>
<td><img src="image10" alt="Synthesized Aerial" /></td>
<td><img src="image11" alt="Top-1 match" /></td>
<td><img src="image12" alt="Ground Truth Aerial" /></td>
<td>532</td>
</tr>
<tr>
<td><img src="image13" alt="Ground Query" /></td>
<td><img src="image14" alt="Synthesized Aerial" /></td>
<td><img src="image15" alt="Top-1 match" /></td>
<td><img src="image16" alt="Ground Truth Aerial" /></td>
<td>1700</td>
</tr>
</tbody>
</table>

Figure 5.6: Some examples of failure cases. The numbers on the right show the position where the ground-truth aerial images were retrieved.

Each other. Also, the matched image has similar color distribution to query image than the ground-truth aerial image. The problem arises because the aerial and ground image pairs in the dataset are captured at different times, so have some visual differences.

Row 3 shows an example where the aerial image has houses which are not captured in street-view images due to occlusion by trees. The impact can also be observed in the corresponding synthesized image which doesn’t contain houses.

Row 4 shows that the street-view image contains a building at far distance. The building covers
large region in ground-truth aerial image, which is difficult to comprehend from the street-view image. Also, this is a rare situation in the dataset with large building. So, the method fails badly, retrieving the ground-truth image at position 1700.

### 5.3.3 Aerial-to-Ground Image Matching: Reverse Problem

We conducted experiments for the reverse problem of Aerial-to-Ground image matching on CVUSA dataset. Here, the aerial image is the query, and we attempt to find the matching ground panorama. First, we use GANs to synthesize ground level panoramas from the aerial images and then use the synthesized images in the proposed joint feature learning and feature fusion methods.

Table 5.3: Image matching performance in terms of Top-1, Top-10 and Top-1% recall on CVUSA Dataset [102] for aerial-to-ground matching.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1</th>
<th>Top-10</th>
<th>Top-1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-stream baseline ((I_g', I_g))</td>
<td>15.04%</td>
<td>37.31%</td>
<td>67.99%</td>
</tr>
<tr>
<td>Two-stream baseline ((I_a, I_g))</td>
<td>16.99%</td>
<td>47.06%</td>
<td>82.11%</td>
</tr>
<tr>
<td>JointFeat. Learning ((I_g', I_g))</td>
<td>16.46%</td>
<td>50.26%</td>
<td>86.26%</td>
</tr>
<tr>
<td>JointFeat. Learning ((I_a, I_g))</td>
<td>27.39%</td>
<td>65.29%</td>
<td>91.46%</td>
</tr>
<tr>
<td>Feature Fusion</td>
<td>44.99%</td>
<td>79.37%</td>
<td>95.66%</td>
</tr>
</tbody>
</table>

We conduct experiments for two-stream baselines, joint feature learning and feature fusion methods. The top-1, top-10 and top-1% accuracies are reported in Table 5.3. We obtain results similar to the numbers reported for ground-to-aerial image matching. We also plot the top-K recall for K = 1 to 80 for the proposed method compared to the baselines in Figure 5.7. This affirms that our method can be applied for image matching in both directions.

The qualitative results for aerial-to-ground image matching are shown in Figure 5.8. The query aerial image, synthesized ground panorama followed by the three closest matches are visualized. The ground-truth panorama are shown with the green borders. We can also observe that the syn-
Figure 5.7: Comparison of our methods with the baselines on CVUSA dataset [102] for reverse problem of aerial-to-ground image matching.

thesized ground panorama are successful in transforming the semantic information from aerial to ground domain.

5.3.4 Cross-view Localization on UCF-OP Dataset

We use the Orlando-Pittsburgh (OP) dataset for image based geo-localization. We want to determine the gps location of the query image by assigning it the location of closest retrieved aerial image. The query image is correctly geo-localized if it is located within a threshold distance in meters from its ground truth position.

We conduct experiments on the OP dataset and provide qualitative results in Figure 5.9. Additional qualitative results are visualized in 5.11. The ground query images are followed by the five closest
The recall accuracy with respect to distance threshold in meters is plotted in Figure 5.10. We
Figure 5.9: Image retrieval examples on the OP dataset. The correct aerial image matches are shown in green borders. The numbers below each aerial image shows its distance in meters from query ground image.

observe that our proposed Feature Fusion method can retrieve images close to its geo-location with higher accuracy than the baseline which can be attributed to its superiority in Top-1 recall.
Figure 5.10: Geo-localization results on the OP dataset with different error thresholds.
Figure 5.11: Cross-view image retrieval examples on the OP Dataset. Ground-truth aerial images are shown in green boxes. The number below each aerial image is its distance in meters from the query image. The first three rows present the images from Orlando and the next three rows of images are from Pittsburgh.
5.4 Summary

In this chapter, we have presented a novel and practical approach to cross-view image retrieval by transforming the query image to target view to obtain a better scene understanding. We showed that the synthesized aerial images can be seamlessly incorporated in cross-view matching pipeline by joint feature training to bridge the domain gap between the aerial and street-view images. Also, the ground image features and the corresponding synthesized aerial image features are fused to obtain a robust descriptor of the ground image. We obtained significant improvements over state-of-the-art methods on the challenging CVUSA dataset and also evaluated on newly collected OP dataset. We solved the geo-localization problem on OP dataset where we had geo-tag information for the reference images. In the next chapter, we propose to extend the idea of image matching to the task of video localization.
CHAPTER 6: VIDEO GEO-LOCALIZATION EMPLOYING GEO-TEMPORAL FEATURE LEARNING AND GPS TRAJECTORY SMOOTHING

The work in this Chapter has been accepted for publication in the following paper:


Chapter 5 dealt with image matching by using deep neural networks to learn the features for the query and gallery images. The best matching gallery image provided the geo-location for the query image. In this chapter, we propose to address the problem of video geo-localization using ground-level driving videos as query. We attempt to determine the GPS location for each frame in the query video. The sequence of GPS locations for the query frames represents the geo-trajectory for the moving camera. Our proposed method learns coherent features for the query frames by exploiting the geographical and temporal proximity between the frames in the video. This helps in estimating smoother trajectories compared to method that learns frame features independently. For reference images, since they are Google Street-View (GSV) images with limited field of view (FOV), there is likelihood of large change in FOV between the consecutive images. The transition between these images is not as smooth as we have for video frames and thus, we impose geographical consistency only in the reference images. We also propose a trajectory smoothing network that learns to identify the outliers in estimated geo-locations and attempts to smooth the trajectories.

The remainder of the chapter is organized as follows: Section 6.1 presents the proposed geo-
temporal feature learning and GPS trajectory smoothing methods in details; Section 6.2 details the dataset used; Section 6.3 discusses the results obtained and Section 6.4 provides the summary of this chapter.

6.1 Methodology

In this work, we leverage the temporal relationships between the frames in a query video to learn their features for the task of video geo-localization. We learn the representations of the query video frames and gallery images differently. The query video frames exhibit temporally smooth transition between consecutive frames, therefore the neighboring frames can be exploited to learn better features for the current frame. The gallery images, on the other hand, are collected from Google StreetView (GSV) and are more discrete and have non-uniform changes between the frames in a trajectory; thus only geographical relationships between the GSV frames is explored. We first obtain embeddings for query and gallery images using encoders, and subsequently improve the feature embeddings by employing transformer based attention networks. Finally, we smooth the estimated geo-locations of the query frames by utilizing a transformer based trajectory smoothing network.

6.1.1 Geo-Temporal Feature Learning Network (GTFL)

The proposed geo-temporal feature learning network is shown in Figure 6.1 and explained in detail next.

**Encoders:** Assume we are given a query BDD video clip of $n$ frames, $B = [b_1, b_2, ..., b_n]$ and corresponding GSV images $G = [g_1, g_2, ..., g_n]$. The encoders, as shown in Figure 6.1, are used to obtain the embeddings for the input frames in our pipeline. The encoders are 2D CNN networks.
with VGG-16 architecture and shared weights. We use the pre-trained weights from the network trained on Pittsburgh 250k dataset [87] and fine-tune them on our dataset. Thus obtained frame embeddings are utilized in the next stage of the pipeline.

Figure 6.1: Geo-Temporal Feature Learning (GTFL) Network: Given a set of \( n \) frames for BDD (query) and corresponding \( n \) frames for GSV (gallery), frame embeddings are obtained using Encoders (VGG-16 network). Then, the Geo-Temporal Attention and Geo-Attention modules learn coherent feature representations for BDD and GSV frame embeddings respectively. The frame-based features are aggregated using Max-pooling operation to obtain the representative clip features, \( b_{clip} \) and \( g_{clip} \). The Frame Triplet loss and GPS loss are applied on frame level features, and the Clip Triplet loss is applied on clip features. Detailed architecture for Attention module is shown on the right where BDD/GSV features are learnt for the frame embeddings.

**Geo-Temporal Attention and Geo-Attention Modules:** The attention modules, Geo-Temporal Attention module (upper branch) and Geo-Attention module (lower branch), have similar architectures, as shown in the right panel in Figure 6.1. The attention module consists of multi-head attention and feed-forward (MLP) layers similar to transformer encoder. We utilize 2 heads and 2
encoders in our attention modules.

The geo-temporal attention module exploits the temporal relationship between the frames in the video clip to learn good frame features. Each feature is learnt by attending to all the frames of the query video. On the other hand, the geo-attention module learns individual features by attending onto itself only. The modules are called ‘Geo-’ modules because the geo-locations of the frames are exploited in learning the features for the query video frames and the reference frames.

Once trained, we utilize the learnt features for estimating the GPS location for each frame in the query clip. We retrieve the matching image feature from the gallery for each query frame feature \([b_{f1}, b_{f2}, ..., b_{fn}]\) and the retrieved features provide the estimated GPS locations for the query video.

### 6.1.2 Loss Functions

We next explain the loss functions used to train the Encoders and the Attention blocks in our proposed architecture. We apply the triplet loss on the frame features as well as on the clip features and additional novel GPS loss to regularize the training.

**Frame Triplet Loss:** Consider the frame features \([b_{f1}, b_{f2}, ..., b_{fn}]\) for a query BDD video with frames \([b_1, b_2, ..., b_n]\), and the feature embeddings \([g_{f1}, g_{f2}, ..., g_{fn}]\) for corresponding matching GSV images \([g_1, g_2, ..., g_n]\). Also, consider the GSV images \([g'_1, g'_2, ..., g'_n]\) from a different location with feature representations as \([g'_{f1}, g'_{f2}, ..., g'_{fn}]\). For the BDD feature \(b_{fi}\), the GSV feature \(g_{fi}\) is a positive feature and the GSV feature \(g'_{fi}\) is the negative feature. Now, if \(d_{pi}\) is the Euclidean distance between the positive feature pairs \((b_{fi}, g_{fi})\) and \(d_{ni}\) is the Euclidean distance between the negative feature pairs \((b_{fi}, g'_{fi})\), the objective of the frame triplet loss is to minimize \(d_{pi}\) as well as maximize \(d_{ni}\). Thus, the frame triplet loss for query clip is computed as the sum of triplet losses for the individual frames of the clip, represented by the Equation 6.1.
\[ L_{\text{triplet-frame}} = \sum_{i=1}^{n} \max(0, m + d_{pi} - d_{ni}), \quad (6.1) \]

where, \( m \) is the margin and \( n \) is the length of the video clip. Triplet loss is a standard loss commonly used in retrieval problems. For image retrieval problems, only the frame feature from current location is a positive feature and the rest are negative features during the training, meaning a different frame from the same clip can be a negative feature. However, in this work we enforce that the negative feature is not of a frame from the same clip.

**Clip Triplet Loss:** As shown in Figure 6.1, we apply max-pooling on the frame features of query video and the set of features for the GSV frames to obtain the representative features \( b_{clip} \) and \( g_{clip} \). We observe that for a small window of 8 frames, the clip frames contain highly overlapping field of views and thus the clip features contain the representative features of the given location. Thus, the clip features for BDD and GSV can additionally be used into the training instead of just employing individual frame features. Therefore, along with the frame triplet loss, we propose to use clip triplet loss to optimize the training.

Assume \( b_{clip} \) and \( g_{clip} \) are BDD and GSV clip features respectively for a given geo-location, their features are considered to be positive feature pairs and their feature distance can be represented as \( d_{p-clip} \). Similarly, if \( g'_{clip} \) is the GSV clip feature at a different geo-location, it is considered as a negative feature for \( b_{clip} \) and the feature distance between \( b_{clip} \) and \( g'_{clip} \) is represented as \( d_{n-clip} \). The clip triplet loss is computed as shown in Equation 6.2.

\[ L_{\text{triplet-clip}} = \max(0, m + d_{p-clip} - d_{n-clip}), \quad (6.2) \]

where, \( m \) represents the margin.
**GPS Loss:** In addition to using frame triplet loss and clip triplet loss, we propose a new loss, GPS loss to further improve the training of our proposed GTFL network. The intuition behind the GPS loss is that the clips (or images) closer to each other in geographical distance are also similar in feature representations compared to the clips (or images) that are further apart in geographical distance. This is because each geographical location may have unique landmarks, landscapes and vegetation representing that region, which can spread over a small nearby area, however, this won’t be valid in regions that are far away. GPS loss acts as an additional supervision to the training since most feature learning is done on image features using triplet losses.

The GPS loss is formulated as explained next. Given GPS info for each frame, we compute the geodesic distance between the frames using the Algorithm for Geodesics [35]. We also compute their feature distances using the learnt feature representations. Let $b_{f1}$ and $b_{f2}$ refer to feature representations for two frames, and let $(lat_1, lon_1)$ and $(lat_2, lon_2)$ be their GPS locations respectively. The geographical distance between two GPS points, $d_{gps}$ is calculated using the algorithm in [35]. Similarly, their feature distance $d_{feat}$ is obtained as shown in Equation 6.3.

$$d_{feat} = || b_{f1} - b_{f2} ||_2^2,$$  \hspace{1cm} (6.3)

We then hypothesize that the normalized feature distance between the frames should be proportional to their normalized geographical distance as shown in Equation 6.4. To verify this, we compute the feature distances $d_{feats}$ for the images and the physical distances $d_{gps}$ for their GPS positions. We visualize these distances in a scatter-plot and fit a line through the points and establish a linear relationship between $d_{feats}$ and $d_{gps}$ with slope 1.077 and intercept of -0.2313; as shown in Figure 6.2.

$$d_{feat} \propto d_{gps},$$  \hspace{1cm} (6.4)
Figure 6.2: Scatterplot showing the relationship between feature distances ($d_{feats}$) and geographical distances ($d_{gps}$) for a subset of the dataset. Each point in the plot represents the feature distance between two images along x-axis and their geographical distance along y-axis. The blue line is the line of best fit through the scatterplot and shows a linear relationship can be established between the points; with a slope of 1.077 and intercept of -0.2313. We model the GPS loss to preserve the linear relationship between the feature distances and gps distances.

We then minimize the $L_1$ distance between the normalized feature distance and the normalized gps distance as shown by Equation 6.5. Any deviation in difference between the feature distance and GPS distance is penalized while training the network.

$$L_{GPS} = || d_{feat} - d_{gps} ||_1,$$  \(6.5\)

We observed that, the inclusion of the GPS loss during the training helps learn discriminative features and contributes to minimizing the localization error.

**Total Loss:** The overall expression for the total loss function is the sum of Equations 6.1, 6.2 and 6.5, as shown in Equation 6.6.

$$L_{total} = L_{triplet-frame} + \lambda_1 * L_{triplet-clip} + \lambda_2 * L_{GPS},$$  \(6.6\)
where, $\lambda_1$ and $\lambda_2$ are the hyperparameters for the loss terms.

Figure 6.3: Proposed Trajectory Smoothing Network: The noisy GPS sequence $[GPS_1, GPS_2, ..., GPS_n]$ is input to the network to compute the error offsets $[\Delta GPS_1, \Delta GPS_2, ..., \Delta GPS_n]$ and the confidence scores $[p_1, p_2, ..., p_n]$ for each input value. The offsets are added to only those GPS values in the input sequence if they are deemed to be noisy by their confidence scores.

6.1.3 Trajectory Smoothing Network

The proposed GTFL network shown in Figure 6.1 is used to obtain the feature representations for the BDD query video frames and the GSV reference images. The geo-location of each query frame is estimated by matching individual frame features to the features of the images in gallery set. The sequence of estimated geo-locations for the query frames represents the trajectory of the moving camera that captured the query video. The predicted trajectory may not be smooth because
even though the query features are learnt jointly, the GPS positions are estimated independently by matching query frame features with the reference image features. Due to some incorrect matches or some outliers, the resultant GPS trajectory may lack temporal smoothness. We thus, propose a trajectory smoothing method to refine the noisy GPS locations. Our approach for temporal smoothing is to determine the noisy GPS values in a set of predicted GPS locations for a query clip. A confidence score along with an offset value for each estimated GPS location is determined such that the addition of the offset to the noisy trajectory will result in a smooth trajectory.

The trajectory smoothing network consists of architecture as shown in Figure 6.3. It consists of linear projection (fc) layer, Transformer encoder layer followed by two parallel heads: a regression head (fc-layer) and a prediction head (fc-layer). The linear projection layer is a fully connected layer that maps a GPS location (2D) to a higher dimensional embedding; a 512 dimensional feature vector. The transformer encoder works on higher dimensional representations for the geo-locations and learns to correct GPS values in the input trajectory. The architecture of the transformer encoder layer is similar to the attention module shown in Figure 6.1, right panel. The learnt embeddings from the transformer encoder are projected back to 2-D space using the regression head. The regressed values represent the normalized values of the offset in GPS error. Also, the learnt embeddings from the transformer are input to the prediction head (fc-layer) that predicts the confidence score whether a GPS location in the input sequence is noisy.

\[
GPS_i' = GPS_i + p_i \Delta GPS_i.
\]

Let \([GPS_1, GPS_2, ..., GPS_n]\) be the predicted geo-locations for the query frames \([b_1, b_2, ..., b_n]\). The estimated geo-locations for majority of the query frames are close to each other, with some possible outliers that account for large errors in localization of the clip. The regression head estimates the offset \([\Delta GPS_1, \Delta GPS_2, ..., \Delta GPS_n]\) for each input GPS values, and the prediction head computes the confidence scores \([p_1, p_2, ..., p_n]\). Depending on the confidence score, the \(\Delta GPS\) is added to
the noisy inputs to obtain the smoothed version of the GPS values; using the Equation 6.7. The confidence score threshold is kept at 0.5.

6.2 Experimental Setup

This section provides details about the datasets used and the implementation details followed in our work.

6.2.1 Datasets

Since there is no existing large dataset to work on video geo-localization problem, we utilize the video clips of BDD dataset [97] provided by Yu et al. as query clips. The BDD dataset is a large-scale driving dataset collected over four different regions of the USA, New York (NY), Berkeley, San Francisco (SF) and Bay Area. The videos are around 40 seconds in length. The dataset provides geo-location (GPS) annotations for the driving trajectories annotated at 1 frame/second. The dataset consists of diverse scene types such as city streets, residential areas and highways. In this work, we consider the BDD video clip as a query, and estimate its corresponding GPS trajectory.

To solve trajectory estimation problem, a reference database of gallery images with known GPS is needed. The feature representations of the query is matched with the gallery image features, and the location of the gallery feature with the highest similarity to the query is selected as the estimated location of the query. Since BDD dataset doesn’t provide the gallery set, we use the GPS annotations of BDD videos to download corresponding Google StreetView (GSV) images at those locations and build a gallery set. For each query location, we download four GSV images, with camera headings of 0, 90, 180 and 270 degrees. We then manually annotate the dataset to

97
select the image that has the highest overlap with the BDD query frames.

Table 6.1: The GPS window of each region under consideration and the area of each region in square kilometers and the number of clips considered from each of the regions from BDD dataset. We employ video sequences only from San Francisco area for training and video sequences from all four areas for testing as shown in the last column of this table.

<table>
<thead>
<tr>
<th>Regions</th>
<th>Latitude Range</th>
<th>Longitude Range</th>
<th>Area  (square kms)</th>
<th># dataset pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Francisco</td>
<td>[37.65, 37.81]</td>
<td>[-122.5, -122.38]</td>
<td>188.06</td>
<td>750 95</td>
</tr>
<tr>
<td>Bay Area</td>
<td>[37.419279, 37.507089]</td>
<td>[-122.258048, -122.1054]</td>
<td>131.69</td>
<td>0 81</td>
</tr>
<tr>
<td>Berkeley</td>
<td>[37.72409913, 37.897474]</td>
<td>[-122.312608, -122.100853]</td>
<td>359.24</td>
<td>0 51</td>
</tr>
<tr>
<td>New York</td>
<td>[40.7073, 40.7381]</td>
<td>[-74.01486, -74.0072]</td>
<td>18.26</td>
<td>0 106</td>
</tr>
</tbody>
</table>

Figure 6.4: The frames from a BDD sequences at times $T = 3, 6, 9, 12, 15$ and 41 (Left Panel, Top); and GSV images corresponding to the same GPS locations (Left Panel, Bottom). The plot of the trajectory (green curve) along with the frame locations (red and blue dots) for different times on the aerial map (Right Panel). Numbers 3, 6, 9, 12, 15 and 41 marked with blue dots on aerial map illustrate the position of moving camera at respective times.

We select different GPS windows for constructing the dataset as shown in Table 6.1. We collect a total of about 750 BDD-GSV pairs (query videos and gallery image) for training and 333 pairs for testing that spreads over 697.25 km$^2$ area. Data from San Francisco area is used for both training and testing, whereas the other three regions, Bay Area, Berkeley and New York are used only for
testing.

A sample sequence of BDD and GSV frames from the dataset is shown in Figure 6.4. The upper row in left panel shows BDD frames at time instances $T = 3, 6, 9, 12, 15$ and $41$ and their corresponding GSV frames are shown in the lower row. We can observe high similarity and overlap in fields of views between the BDD and GSV frames; justifying that we are successful in constructing a meaningful dataset employing BDD video frames and GSV images. The right panel shows the camera location at different time instances with (red and blue) dots on an aerial map; and the green curve connecting them demonstrates the path that the camera takes.

6.2.2 Implementation Details

In this subsection, we present the implementation details of our transformer based architecture for geo-temporal feature learning and trajectory smoothing networks. We use PyTorch [60] for the implementations.

Geo-Temporal Feature Learning Network: The GTFL network consists of encoders followed by attention modules as shown in Figure 6.1. The encoders are VGG-16 networks, with NetVLAD layer as the final layer, and shared weights for both branches. We use the pretrained weights of the network trained on Pittsburgh 250k dataset to initialize the parameters of the network and fine-tune them on our dataset. The output of frame-encoder network is a 32,768-dimensional feature representation for each input frame. Geo-temporal attention and geo-attention modules consist of encoder modules of Transformer network. We use 2 attention heads and 2 encoder layers for both modules. The weights are randomly initialized.

We use triplet losses on the frame features, and the clip features as well as the proposed GPS loss on the frame features. The $\lambda_1$ and $\lambda_2$ are the balancing factors between the losses and their values are
set at 10 and 10 respectively. The query and gallery features are represented by 512-dimensional vectors.

**Trajectory Smoothing Network:** The trajectory smoothing network consists of a fully-connected layer that maps 2-dimensional GPS values to 512 dimensional representations, followed by the encoder module of transformer. The transformer encoder consists of four attention heads and two encoder layers. The output of the transformer encoder is passed through two parallel heads, fully connected layers that map the 512-dimensional representations to 2-dimensional values of offset regression and confidence score prediction. During the training, we employ data augmentation by feeding noisy GPS values and artificially perturbed GPS values as input to the network. We observe that by artificially perturbing some ground truth GPS values and using them as input provides the network with strong guidance that not all GPS location are noisy and only some need modifications, whereas the rest should be kept unchanged.

### 6.3 Results

We present extensive evaluation of our proposed method demonstrating the effectiveness of collectively learning the features for the clip to estimate a smoother trajectories for the query videos.

#### 6.3.1 Evaluation Metrics

We provide evaluation in terms of localization error as well as recall accuracy. For localization error, we first compute the distance in meters between the estimated GPS positions of the query video frames and their ground truth GPS locations. Average of the error distances for the frames provides the localization error for the query clip.
Recall accuracy is reported in terms of recall at top-K and recall at distance threshold. For recall accuracy at top-K, a matching is successful if the correct match is within a set of K closest images in Euclidean distance of their features. For recall accuracy at distance threshold, a query is correctly localized if its distance in meters to its ground truth position is within the threshold distance.

6.3.2 Quantitative Evaluation

We present the quantitative evaluation of the baselines and our proposed method in terms of localization error and recall accuracy.

Localization Error: We compare our proposed approach with the baseline 2D CNN and 3D CNN architectures. The baseline networks are explained next.

The 2D CNN baseline consists of VGG-16 architecture with NetVLAD as final layer, similar to encoder of GTFL network and uses the pretrained weights as explained in Section 6.1.1. We conduct the evaluation using the raw features obtained using the pretrained weights and report the results in first row of Table 6.2. We next finetune the 2D CNN baseline on our dataset. The features for each frame in the query video clip are learnt independently and their GPS locations are predicted using the query features. The predicted locations are smoothed using the proposed smoothing network. The results are presented in the second row of Table 6.2.

We also conduct the baseline experiment using 3D CNN architecture to learn the feature embeddings for the query frames. Here, the ResNet R3D-18 is used with the modification that the temporal dimension is preserved; meaning the output for N input frames in a clip will have N features. But these features are learnt by considering the neighboring frames as well, since the kernel size of 3 for temporal dimension is considered. The results are presented in the third row of Table 6.2. Since the network is trained from scratch, it performs worse compared to 2D CNN baseline.
Table 6.2: Comparison of proposed approach with baseline methods in terms of localization error (meters). 2D CNN: Evaluation using raw features from pretrained VGG network. 2D CNN\(^f\): VGG network fine-tuned on our dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>SF</th>
<th>Bay Area</th>
<th>Berkeley</th>
<th>NY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D CNN</td>
<td>2516</td>
<td>4686</td>
<td>7020</td>
<td>1818</td>
</tr>
<tr>
<td>2D CNN + Smoothing</td>
<td>2290</td>
<td>3999.17</td>
<td>5425</td>
<td>1292</td>
</tr>
<tr>
<td>2D CNN(^f)</td>
<td>2091.66</td>
<td>4509.15</td>
<td>6687.61</td>
<td>1332.08</td>
</tr>
<tr>
<td>2D CNN(^f) + Smoothing</td>
<td>1710.54</td>
<td>4112.15</td>
<td>4565.46</td>
<td>1222.88</td>
</tr>
<tr>
<td>3D CNN</td>
<td>4247.09</td>
<td>6183.71</td>
<td>6677.93</td>
<td>1572</td>
</tr>
<tr>
<td>3D CNN + Smoothing</td>
<td>3848.83</td>
<td>5201.65</td>
<td>6503.96</td>
<td>1399.17</td>
</tr>
<tr>
<td>Proposed</td>
<td>300.47</td>
<td>524.28</td>
<td>424.79</td>
<td>493.43</td>
</tr>
<tr>
<td>Proposed + Smoothing</td>
<td>128.94</td>
<td>206.32</td>
<td>161.51</td>
<td>285.43</td>
</tr>
</tbody>
</table>

Finally, we present the results for the proposed method in the fourth row of Table 6.2. For our proposed method where the network is able to consider all the frames in the input to generate their individual features, the results are significantly better than the networks that use 2D CNN and 3D CNN baseline architectures only. Also, smoothing of the trajectory helps to reduce the localization error even further.

**Recall Accuracy:** We next report the comparison of our proposed method with the best performing baseline method (2D-CNN) in terms of recall accuracies. We present the top-K recall accuracy for K = 1 to 100 in Figure 6.5a. Here, we visualize the recall accuracy plot for all four regions. We observe that our proposed method performs significantly better than the baseline method. We also report recall accuracy with respect to distance threshold in Figure 6.5b. We observe that the proposed method is better than the baseline for all values of distance thresholds. At a distance threshold of 200 meters, the recall accuracy for the proposed method and baseline 2D-CNN are 82.35% and 19.52% respectively.
Figure 6.5: Comparison of the proposed approach with baseline 2D-CNN in terms of recall accuracies.

The recall accuracy plots illustrate the superiority of our proposed approach to video geo-location compared to the frame based baseline network.

### 6.3.3 Qualitative Evaluation

Figure ?? shows the geo-spatial trajectories predicted by our proposed method on subset of videos from San Francisco Area and their comparison with the ground truth trajectories. The green curves represent the ground truth trajectories for the camera while the red curves are the trajectories predicted by our proposed method. These qualitative results demonstrate the capability of the proposed method in large-scale video localization.

Next, we provide additional qualitative results for all four regions of evaluations, San Francisco, Berkeley, Bay Area and New York. Figure 6.6 visualizes three sample images from Bay Area (first row) and San Francisco (second row). Similarly, Figure 6.7 shows sample images from New York (first row) and Berkeley (second row). The green curves represent the ground truth trajectories and
Figure 6.6: Qualitative Results. Example Images showing the ground truth (green curves) and predicted (red curves) trajectories for Bay Area (first row) and San Francisco (second row). The red curves show the corresponding predicted trajectories in each image. We can observe that the predicted trajectories have a very high overlap with the ground truth trajectories, justifying that the network is able to localize the trajectories successfully.
Figure 6.7: Qualitative Results. Example Images showing the ground truth (green curves) and predicted (red curves) trajectories for New York (first row) and Berkeley (second row).

We also present a trajectory smoothing example in Figure 6.8. Figure 6.8a presents a noisy trajectory (blue curve) obtained by using the proposed GTFL network. The trajectory smoothing network refines the noisy trajectory resulting in the smooth trajectory as shown in Figure 6.8b. We observe significant impact of trajectory smoothing network in obtaining smoother predicted trajectory.
Figure 6.8: Trajectory smoothing example for a query clip from Berkeley region. (a) shows the ground truth (green curves) and the predicted trajectories before smoothing (blue curves). (b) shows the ground truth (green curves) and the predicted smooth trajectories (red curves).

Table 6.3: Ablation study of GPS errors in meters with respect to different losses during the training. TL$_f$: Frame Triplet loss; TL$_c$: Clip Triplet loss; GL: GPS Loss.

<table>
<thead>
<tr>
<th>Losses</th>
<th>SF</th>
<th>Bay Area</th>
<th>Berkeley</th>
<th>NY</th>
</tr>
</thead>
<tbody>
<tr>
<td>TL$_f$</td>
<td>704.16</td>
<td>1172.99</td>
<td>1253.6</td>
<td>978.51</td>
</tr>
<tr>
<td>TL$_f$ + TL$_c$</td>
<td>524.81</td>
<td>692.34</td>
<td>819.54</td>
<td>836.13</td>
</tr>
<tr>
<td>TL$_f$ + TL$_c$ + GL</td>
<td>300.47</td>
<td>524.28</td>
<td>424.79</td>
<td>493.43</td>
</tr>
</tbody>
</table>

6.3.4 Ablation Study

We conduct ablation studies to understand the impact of different loss functions used in our experiments. We also conduct ablations on the parameters of trajectory smoothing network to determine the best hyperparameters. Next, we present the ablation on output feature dimensions of the GTFL network. Finally, we conduct experiments with and without NetVLAD layer in our network to understand the contribution of NetVLAD layer.
Ablation on Losses: For this ablation, we conduct the experiments with different combinations of loss functions for our proposed method. The results after the application of the trajectory smoothing on the predicted GPS are shown. The results are presented in Table 6.3. The numbers suggest that utilizing the clip triplet loss helps in obtaining better GPS estimation compared to with only frame triplet loss; and the use of GPS loss further helps the network to learn discriminative features for the clips and the localization error decreases further.

Table 6.4: Ablation on the number of heads in self-attention module and the number of transformer encoder layers.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>SF</th>
<th>Bay Area</th>
<th>Berkeley</th>
<th>NY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heads = 2, Layers = 1</td>
<td>158.07</td>
<td>233.86</td>
<td>189.54</td>
<td>307.91</td>
</tr>
<tr>
<td>Heads = 2, Layers = 2</td>
<td>144.21</td>
<td>211.59</td>
<td>168.61</td>
<td>292.24</td>
</tr>
<tr>
<td><strong>Heads = 4, Layers = 2</strong></td>
<td><strong>128.94</strong></td>
<td><strong>206.32</strong></td>
<td><strong>161.51</strong></td>
<td><strong>285.41</strong></td>
</tr>
<tr>
<td>Heads = 4, Layers = 4</td>
<td>129.46</td>
<td>227.78</td>
<td>183.97</td>
<td>312.57</td>
</tr>
</tbody>
</table>

Ablation on parameters for Trajectory Smoothing Network: For this ablation, we conduct experiments for smoothing the predicted GPS trajectory by varying the number of layers of transformer encoder network and varying the number of heads in the self-attention layer. The result is shown in Table 6.4. We observe that the best results are obtained for heads = 4 and layers = 2.

Ablation on Feature Dimensions: Feature dimensions are critical components in deep learning networks. The feature dimension represents the size of a feature vector representing each input image. Larger feature dimension means larger memory requirements to store them as well as more computations. Smaller dimension provides more compact representations but they may be insufficient for mapping the images to feature space.

We conducted experiments by varying the feature dimensions as 256, 512 and 1024 and report the results in Table 6.5. As observed, 512 dimensional feature representation works the best for our experiments.
Table 6.5: Ablation study on varying feature dimension size. We report the evaluations in terms of localization error in meters.

<table>
<thead>
<tr>
<th>Feature dim.</th>
<th>SF</th>
<th>Bay Area</th>
<th>Berkeley</th>
<th>NY</th>
</tr>
</thead>
<tbody>
<tr>
<td>d=1024</td>
<td>472.46</td>
<td>642.17</td>
<td>507.03</td>
<td>603.72</td>
</tr>
<tr>
<td>d=512</td>
<td>300.47</td>
<td>524.28</td>
<td>424.79</td>
<td>493.43</td>
</tr>
<tr>
<td>d=256</td>
<td>363.24</td>
<td>888.99</td>
<td>579.49</td>
<td>518.47</td>
</tr>
</tbody>
</table>

**Ablation with and without NetVLAD layer:** NetVLAD [2] is a popular trainable pooling layer used to capture the information about the statistics of local descriptors aggregated over the image. NetVLAD learns the cluster centers and residuals. NetVLAD has been widely used in image retrieval problems and thus we use it in our framework as well. Here, for this ablation, we conduct experiments with and without NetVLAD layer and present the results in Table 6.6. We can observe that the network with NetVLAD layer performs slightly better than the network without NetVLAD layer. This affirms that NetVLAD helps in retrieval problems; but the large improvement in results for our proposed method over 2D CNN (as reported in Table 2 in the main paper) is contributed by temporally learning of features and not necessarily due to the use of NetVLAD in our network.

Table 6.6: Ablation study on experiments with and without NetVLAD layer in the proposed network. We report the evaluations in terms of geo-localization error in meters.

<table>
<thead>
<tr>
<th>Methods</th>
<th>SF</th>
<th>Bay Area</th>
<th>Berkeley</th>
<th>NY</th>
</tr>
</thead>
<tbody>
<tr>
<td>without NetVLAD</td>
<td>531.53</td>
<td>655.49</td>
<td>736.41</td>
<td>715.98</td>
</tr>
<tr>
<td>with NetVLAD</td>
<td>300.47</td>
<td>524.28</td>
<td>424.79</td>
<td>493.43</td>
</tr>
</tbody>
</table>
6.3.5 Feature Visualization

In Figure 6.9, we visualize the BDD (query) and GSV (gallery) image features obtained from baseline 2D CNN and proposed GTFL methods. The feature representation for each frame is converted to a two-dimensional embedding using t-SNE [50] for visualization. The red and cyan circles close to each other represent the features for query and gallery frame pairs.

We observe that the scatter-plot for features obtained using 2D CNN has less overlap between BDD and GSV features compared to the scatter-plot for proposed method. The proposed method is successful in bringing the ground-truth GSV features closer to BDD features and pushing the non-matching features away, compared to the baseline. Thus, the feature clusters are more compact for proposed method compared to the baseline. The clusters represent the features for frames belonging to the same trajectory. Geo-temporal feature learning helps obtain smooth features for frames of the same clip as can be observed on the right plot.
6.4 Summary

In this chapter, we have presented a novel application of transformer based networks for long-term feature learning between the frames of a query video clip for the task of video geo-localization as well as for geo-trajectory smoothing. We formulated novel GPS loss and validated its contribution in learning better features for the query and gallery frames. We built a new benchmark dataset for video geo-localization and report significant improvement of proposed method over frame based feature learning approach where the temporal relations between the frames are not captured and over 3D-CNN baseline where only short term temporal information is incorporated.
CHAPTER 7: CONCLUSION AND FUTURE WORK

Here, we highlight the concluding remarks and provide insights into the future research works in this area.

7.1 Final Conclusions

In Chapter 3, we introduced the problem of cross-view image synthesis and proposed Generative Adversarial Network based architecture to learn relationships between the semantics in the two views and generated realistic cross-view images. We proposed X-Fork and X-Seq architectures that synthesize the cross-view semantic segmentation maps as an auxiliary output of the network and helps to regularize the network. We validated the effectiveness of the proposed architectures against baseline approaches that do not learn additional cues (e.g., semantic segmentation maps) using different evaluation metrics.

In Chapter 4, we explored the geometric relationship, i.e., homography between the aerial and the ground images to project the aerial images to ground level perspective and use them as input to our cross-view synthesis architectures proposed in Chapter 3. This guides the network to emphasize on learning to synthesize sharper images and focus less on the viewpoint transformation. We conducted extensive evaluations using various evaluation metrics showing compelling evidence of the superiority of the proposed approach.

In Chapter 5, we leveraged our cross-view image synthesis work to bridge the domain gap between the ground and aerial images for the task of image matching. Given a ground query image, we synthesized cross-view (aerial) images using X-Fork architecture and used the synthesized images as additional cues to learn better query features. We demonstrated that it is possible to utilize the
synthesized images to minimize the domain gap between the query and gallery viewpoints and the extensive evaluations validated the superiority of proposed method over the state-of-the-art methods for the task of cross-view image matching.

In Chapter 6, we continued the exploration of image geo-localization to videos. For video geo-localization, the geo-location of each frame in the video is estimated and the path represented by these geo-locations represents the trajectory of the moving camera that recorded the video. We proposed novel architecture to learn temporally and geographically coherent features for the frames in a video clip and demonstrated superiority of proposed approach over the baseline methods. We also proposed novel architecture to determine noisy GPS values in the estimated trajectory and regressed the offset values for those noisy GPS points such that the addition of these offsets to the noisy GPS values estimated the smooth trajectory. We constructed a large-scale benchmark dataset for video geo-localization problem and conducted extensive evaluations to demonstrate the success of proposed method in solving the problem.

7.2 Future Research Directions

As with most research works, we believe that there are areas of improvements and extensions in our research works.

In Chapter 3 and 4, the proposed networks learned the relationship between the limited FOV ground images and the corresponding aerial images from the dataset. This was possible because the cross-view images are aligned. Generating an aerial image from such ground images would expect the network to hallucinate large parts of images that may look pleasing to the eyes but not a correct depiction of the location. Future researchers can explore mechanisms to establish stronger correspondences between the overlapping FOV between the viewpoints for a more meaningful
synthesis. Similarly, cross-view image synthesis can be extended to videos. Cross-view video synthesis can be interesting research direction where ground level videos can provide the conditioning parameter to synthesize the aerial videos and vice versa.

In Chapter 5, the images synthesized from ground level panorama provided additional cues for the query image. It is not yet explored how the synthesized images would contribute in learning features when the query images are of limited (90 degree) field of view. This could be useful in geo-localizing the video frames since the missing field of view can be compensated by leveraging the temporal evolution of the videos. Additionally, future research direction include exploring the orientation relationship between the unaligned ground and aerial images. This can be done by utilizing the synthesized images as cue to determine the orientation offset between the cross-view images. This would be particularly practical when we have randomly oriented ground level images while the aerial images are aligned with north up. This should facilitate in trajectory estimation of videos particularly at junctions where the camera heading is changing in the video.

For video trajectory estimation problem, future direction include utilizing the aerial images as gallery that can help build a standard gallery set with full coverage for a region of interest. Researchers can explore ways of better feature learning, especially on handling transient objects like cars, person, etc. in the query frames. This can be done by either masking these objects as a data augmentation step during the training; or by regularizing the network training by weighing differently on loss components for each segmentation class. Also, efforts can be put to determine effective ways to smooth the noisy trajectories.
LIST OF REFERENCES


