TEC | Tecnológico de Costa Rica

Unsupervised Optical Flow With Globally Optimized Cost Volume

Thesis in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Author: Jennifer Caballero Solís

Advisor: Francisco J. Torres Rojas, Ph.D.

June 13, 2024

Escuela de Computación
Unidad de Posgrado

TEC | Tecnológico de Costa Rica

ACTA DE APROBACION DE TESIS

Unsupervised Optical Flow With Globally Optimized Cost Volume

Por: Jennifer Caballero Solís

TRIBUNAL EXAMINADOR

Dr. Francisco Torres Rojas
Profesor Asesor

MSc. Kervin Sánchez Herrera
Profesor Lector

MSc. Marco Madrigal Solano
Lector Externo

Dra.-Ing. Ulliana Sancho Chavarría
Presidente, Tribunal Evaluador Tesis
Programa Maestría en Computación

SINAES POST-GRADO ACREDITADO COSTA RICA

13 de junio, 2024

Abstract

A wide variety of applications rely on computer vision to recover properties of the 3D world from 2D input images. Optical flow is a fundamental task for applications that require motion estimation. Large displacements and repetitive patterns in the images make optical flow estimation a challenging problem.

Deep learning has pushed the state-of-the-art on various computer vision tasks, including optical flow estimation. To our knowledge, the supervised learning models have achieved the best performance in optical flow estimation. Supervised learning requires large datasets with ground truth for training, but real image datasets for optical flow are scarce and hard to generate, consequently models are trained with synthetic images instead.

Datasets with real images for unsupervised learning of optical flow estimation are abundant and easy to obtain, as ground truth is not required. Images of the same domain can be used for inference and training of the optical flow estimation model with unsupervised learning. For these reasons, unsupervised learning of optical flow estimation is an interesting research field.

In this study, we investigate two potential enhancements for the UFlow model, an unsupervised learning optical flow estimation model, with the aim of achieving improved accuracy in scenarios involving large displacements (40 pixels or more). Our focus centers on refining the performance of the model by addressing the cost volume layer.

Our contributions are:

1. The implementation of global cost volume computation for UFlow and its evaluation in the two higher levels of the feature pyramid.

2. The integration of globally optimized cost volume (GOCor) with UFlow and its evaluation.

3. The evaluation of the combined effects of global cost volume and GOCor.

We discovered that the usage of global cost volume and GOCor increases significantly the computational requirements to train and infer the optical flow, namely the GPU memory required. The results obtained indicate that the concurrent use of GOCor and global cost volume does not yield gains in the optical flow estimation for large displacements, but the sole use of GOCor does.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer vision aims to recover physical scene properties from images. Shape, illumination, and color distributions are used to describe and reconstruct the world we see [25]. The estimation of optical flow from images is a fundamental problem in computer vision and also an essential capability of any robotic system [11]. Applications that use optical flow for motion analysis include action recognition [20], video encoding [9], visual odometry, depth estimation [33], structure from motion [19], and object segmentation and tracking [18]. Given that optical flow is a fundamental building block in computer vision, improvements in its estimation benefits many applications [8].

From the seminal work of Horn and Schunk 40 years ago [5], optical flow estimation has been an active research field with steady progress [23]. When a 3D scene is projected into the 2D image plane, depth and other information from the 3D scene is lost, causing ambiguity in computer vision tasks [3]. In addition, occlusions, illumination changes, and large displacements are among the challenges in optical flow estimation [29]. Fast and large motion of the camera and scene can lead to large displacements of pixels between consecutive frames, an usual problem in many applications like advanced driving assistance systems (ADAS) [17].

In recent years, the performance of optical flow estimation has improved dramatically using deep learning techniques [8]. In some cases, the deep learning approach has set a new state-of-the-art, providing better results than traditional computer vision approaches in tasks like image classification, such as AlexNet in 2012 [10].

In 2015, FlowNet [4] demonstrated that optical flow could be estimated using convolutional neural networks (CNNs) and supervised learning. Although supervised learning methods to estimate optical flow outperform unsupervised, the second is an interesting field of research because of the availability of the datasets required. Supervised training needs large datasets with ground truth, however, these are scarce and difficult to generate from real image videos, thus previous works have used synthetic images datasets [26]. In contrast, real image sequences for training unsupervised optical flow estimation are abundant and easy to generate [8] since it does not require labeled ground truth. For applications that perform inference of optical flow from real image sequences, relying on networks trained with synthetic datasets causes domain mismatch [8]. This problem can be circumvented in unsupervised learning models.

1.0.1 Problem Statement

The cost volume has become a fundamental block of deep learning models in computer vision [28] to establish a correlation between image feature maps (extracted using CNNs). Often, the cost volume is computed locally to reduce the computational load required during training and inference. In the local cost volume computation, each feature map component is correlated with only a small neighborhood of the second feature map. If the correspondence between the features is out of the search window of the second feature map due to large displacements, then the cost volume will fail to register the similarity. If the cost volume is computed globally, besides the additional computational load, repetitive patterns in the images can cause am-

biguous confidences affecting the final optical flow estimation. As repetitive patterns are usually present in both input images, a model could leverage this information to filter the incorrect correspondences improving the cost volume output, as proposed in GOCor [28].

The cost volume layer is used in both supervised and unsupervised models, however, in this work we are interested in unsupervised models.

To push the state-of-the-art in unsupervised learning of optical flow, we address the problem of large displacements, improving the cost volume layer. This research evaluates two key optimizations to the cost volume layer of an unsupervised learning model: replacing local cost volume by global cost volume in the higher levels of the feature pyramid and using the GOCor module to replace the traditional computation of the cost volume. Our study aims to contribute to the optical flow estimation for large displacements, which is a common problem in many use-cases. Our research has the potential enhance the results of other computer vision applications employing optical flow and conducting inference on non-synthetic images.

1.0.2 Outline

Chapter 2 reviews the background knowledge on optical flow and models for learning optical flow. In chapter 3, we detail the research hypothesis and the objectives. Chapter 4 presents the methodology used for this research. In chapter 5, we provide the research results and analysis. Chapter 6 presents the conclusions of this study.

Chapter 2

Background

2.1 Optical Flow

The American psychologist James Gibson introduced the concept of optical flow in his study of the visual perception of space in 1940, he argued that the retinal images perceived through the eyes are related to the motion and the changes in gradients of light [15]. In 1981, Horn and Schunck [5] defined optical flow as *"the distribution of apparent velocities of movement of brightness patterns in an image"*.

Consider two consecutive images, I_1 captured at $t = 1$ and I_2 captured at $t = 2$ as depicted in the figure 2.1. In this scenario, the color pattern moved between the images due to camera motion, scene motion, or both. For each pixel in I_1, optical flow estimation attempts to find its new location in I_2. The estimation of optical flow assumes brightness constancy -that the brightness of the corresponding pixels is the same in both images-, and that the motion is small.

Brightness constancy and smoothness (small motions) are two core assumptions for optical flow estimation [5]. Equation 2.1 captures the brightness assumption, where (x, y) is the pixel position in I_t and (u, v) is the flow vector between I_1 and I_2.

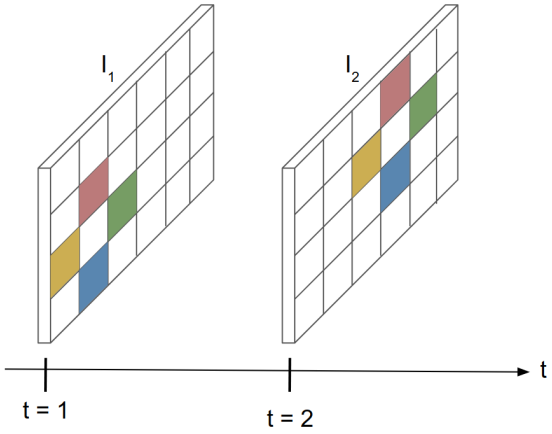

Figure 2.1: Consecutive images

$$I_1(x+u, y+v) - I_2(x, y) = 0 \tag{2.1}$$

Figure 2.2 shows an example of optical flow estimation, where the 2D image plane origin is on the top left corner of the image, and the same brightness pattern observed in I_1 has moved two pixels in the x axis positive direction and one pixel in the y axis negative direction in I_2. The optical flow field contains the 2D motion vectors (u, v) for all the image pixels.

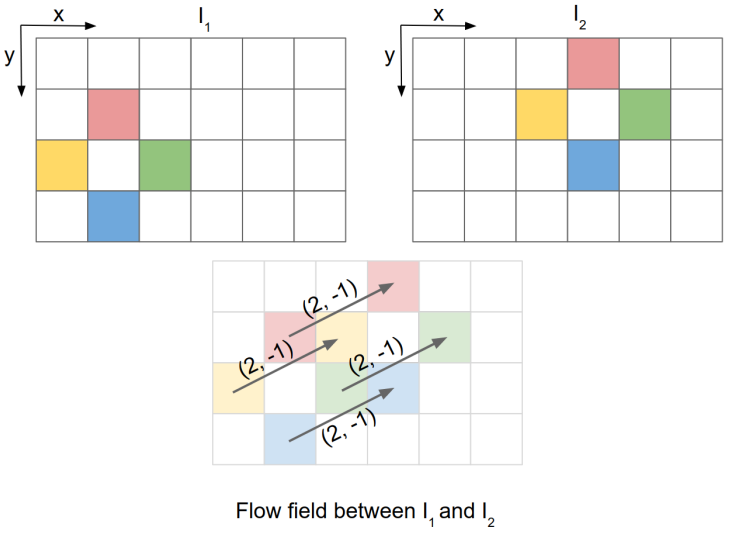

Figure 2.2: Optical flow field

Optical flow estimation is an underconstrained problem [25], it attempts to recover

two values (u and v) with one equation.

The optical flow estimation field research has shown steady progress [23] since Horn and Schunck's original formulation in 1981 [5]. More recently the use of convolutional neural networks (CNNs) to estimate optical flow has outperformed former techniques [4].

To our knowledge, the best performing model for optical flow estimation to date [21] uses supervised learning. Supervised learning schemes to train CNNs depend on the availability of large datasets annotated with ground truth [32]. Labeled datasets for optical flow are scarce and difficult to generate for real scenes which leads to the use of computer generated synthetic datasets [8].

On the other hand, unsupervised learning approaches can count on abundant data as they are trained to optimize a proxy objective [8] instead of depending on the ground truth information. The subsequent sections offer information about the supervised and unsupervised training approaches for optical flow estimation, delve into previous work on enhancements to the cost volume layer and present details about the datasets available for evaluating optical flow estimation.

2.2 Datasets

Among the most popular datasets for optical flow training and evaluation are KITTI 2015 [16], Sintel [2], and Flying Chairs [4]. KITTI 2015 is a dataset focused on autonomous driving applications that augmented semi-dense flow fields using disparity maps from 3D information obtained with laser scans [16]. Sintel is a computer generated dataset that contains large motions, blur, atmosferic effects, and other components that challenge optical flow estimation [2]. Flying Chairs is a synthetic dataset that uses chairs rendered on top of random images [4].

Typically, the endpoint error (EPE) serves as a metric to assess the quality of an

optical flow estimate. The endpoint error is calculated by averaging the Euclidean distance between the ground truth optical flow and the estimation for each pixel across all image pixels. [35]. EPE provides information on how close is the estimated optical flow to the ground truth.

2.3 Supervised Learning of Optical Flow Estimation

In 2015, a CNN based model called FlowNet [4] used supervised learning of optical flow to achieve better results than state-of-the-art methods in the Flying Chairs dataset. FlowNet architecture was trained end-to-end as shown in figure 2.3.

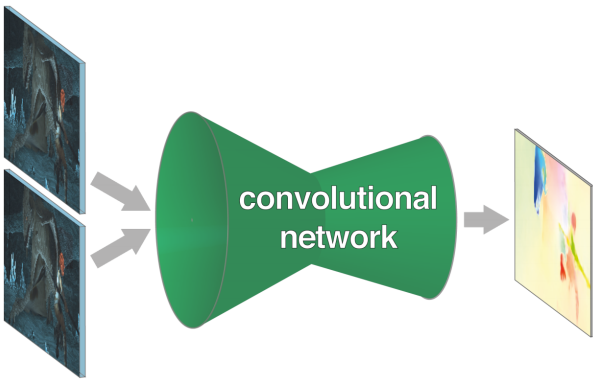

Figure 2.3: End-to-end optical flow training
Source: [4]

Previous work successfully addressed the problem of learning features from images with CNNs, such as AlexNet in 2012 [10]. However, optical flow estimation requires feature matching in addition to feature extraction. In 2015, FlowNet pioneered the utilization of CNNs for feature matching [4].

In 2018, Sun *et al.* combined multi-scale (pyramidal) approach, image warping, and cost volume, also called domain knowledge techniques, in a CNN-based model that achieved state-of-the-art results for optical flow estimation trained with supervised learning. This model called PWC-Net [24] had fewer hyperparameters than other

state-of-the-art models and similar accuracy. The following subsections provide more detail about the domain techniques integrated by PWC-Net and its architecture.

2.3.1 Multi-scale (Pyramid)

In images with large displacements, the optical flow can be estimated and refined at different resolutions, this is called coarse to fine (multi-resolution) estimation [23]. Image pyramids can be built halving their resolution, at each level, the downscaled image has a quarter of the pixels than the previous level [25], as shown in figure 2.4. In a pyramidal approach, the optical flow is estimated first at the higher pyramid level that has the smallest resolution and then refined in the lower levels with higher resolutions.

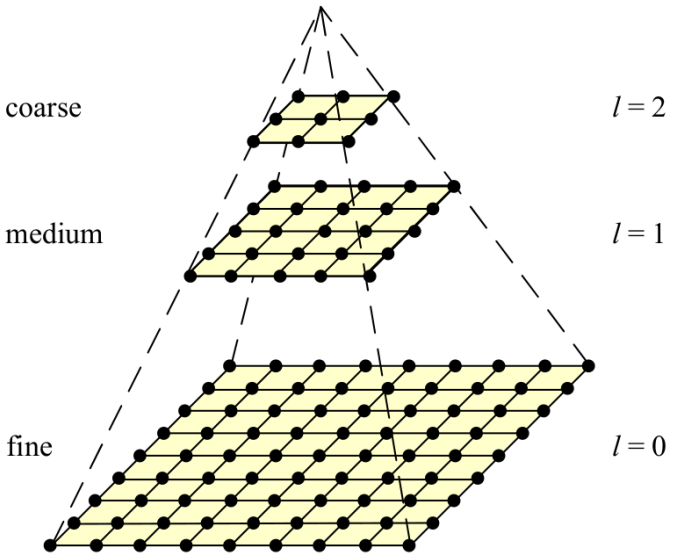

Figure 2.4: Image pyramid
Source: [25]

2.3.2 Image Warping

Image warping deals with geometric transformations that redefine the spatial relationship between the points in the image [30]. Each pixel from the input image $f(x, y)$ located at (x, y) is mapped to a different position determined by the transfor-

mation $h(x, y)$, as in equation 2.2. A transformation example is shown in the figure 2.5, where the red and green pixels are mapped to different positions according to $h(x, y)$. Rotation, scaling and translation are common transformations in image processing.

$$g(x, y) = f(h(x, y)) \tag{2.2}$$

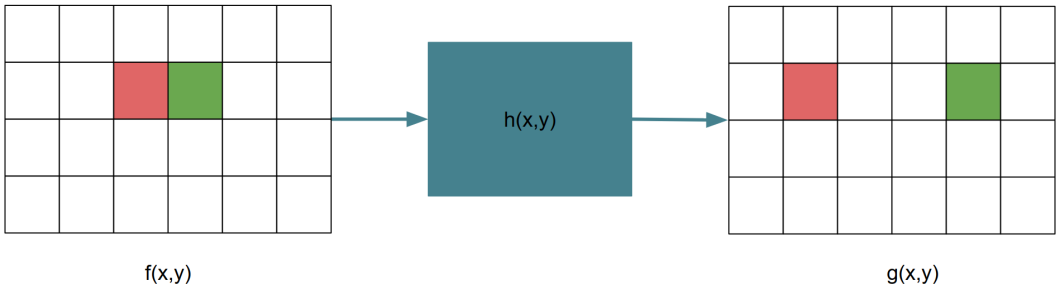

Figure 2.5: Example of an image geometric transformation

In the context of optical flow, warping is used to apply the estimated optical flow between the images. Consider two consecutive input images $I_1(x, y)$ and $I_2(x, y)$, each of size WxH, and the estimated forward dense flow field between these images $F_{1,2}(x, y)$ of size $WxHx2$. Then for each pixel of I_1 located at (x, y) there is one vector in $F_{1,2}(x, y)$ that describes the motion of that pixel with respect to I_2 and the flow field $F_{1,2}(x, y)$ can be used to warp I_2 toward I_1 and evaluate how much the warped image resembles I_1.

2.3.3 Cost Volume

A cost volume is built by matching two feature maps F_1 and F_2 and storing the cost for associating each pixel in F_1 to each pixel in F_2 [24]. CNN-based feature extractors generate several maps per image. Consider the figure 2.6, each CNN extracts d feature maps from each input image, and the pixel value of the maps at (x, y) is a vector of size d (each component represents the result of the filters applied

by the CNN centered in that pixel). The global cost volume is the scalar product of each vector of F_1 at (i, j) with each vector of F_2 at (m, n) where $0 \leq i, m < W$, $0 \leq j, n < H$ and $0 \leq k < D$ with $D = W \times H$, according to equation 2.3 [28]. Algorithm 1 shows the pseudocode to compute the global cost volume, a five level nested loop with a computational complexity $O(H^2W^2d)$.

$$C(i, j, k) = (F_1(i, j))^T F_2(m, n) \tag{2.3}$$

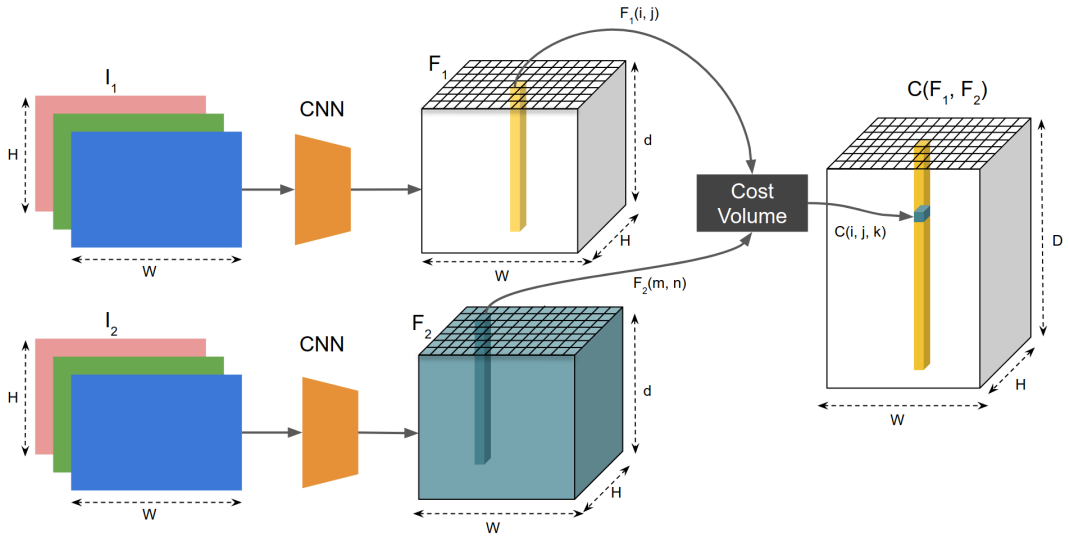

Figure 2.6: Global cost volume computation

Algorithm 1: Global cost volume

```
for j ← 0 to H do
    for i ← 0 to W do
        k ← 0;
        for n ← 0 to H do
            for m ← 0 to W do
                for p ← 0 to d do
                    C(i, j, k)+ = F1(i, j, p) ∗ F2(m, n, p);
                end
                k+ = 1
            end
            k+ = 1
        end
    end
end
```

The cost volume can also be computed for a neighborhood of pixels instead of the whole size of the F_2 feature maps, this is called local cost volume and is computed with equation 2.3 as well, but in this case $D = S^2$ [28] where S is the size of search neighborhood as illustrated in image 2.7. This approach, also called local cost volume, requires less computational resources than the global approach, but may not detect large displacements if the correspondence is out of the search neighborhood. Algorithm 2 shows the pseudocode to compute the local cost volume, with a computational complexity $O(HWS^2d)$.

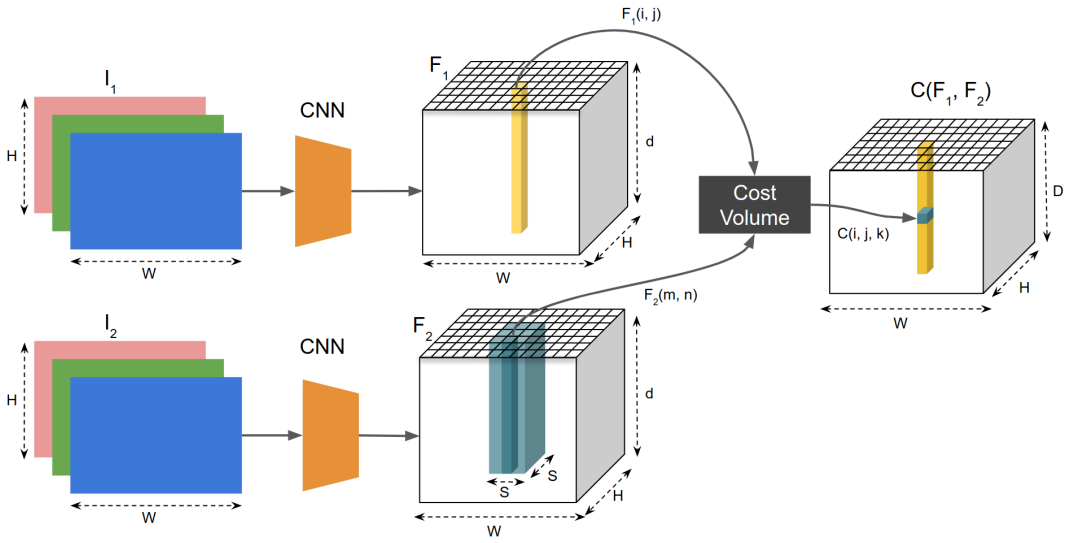

Figure 2.7: Local cost volume computation

Algorithm 2: Local cost volume

for $j \leftarrow 0$ **to** H **do**
  for $i \leftarrow 0$ **to** W **do**
    $k \leftarrow 0$;
    for $n \leftarrow (j - \lfloor \frac{S}{2} \rfloor)$ **to** $(j + \lfloor \frac{S}{2} \rfloor)$ **do**
      for $m \leftarrow (i - \lfloor \frac{S}{2} \rfloor)$ **to** $(i + \lfloor \frac{S}{2} \rfloor)$ **do**
        for $p \leftarrow 0$ **to** d **do**
          $C(i, j, k) += F_1(i, j, p) * F_2(m, n, p)$;
        end
        $k += 1$;
      end
      $k += 1$;
    end
  end
end

[HTML]409DAB Pyramid level (l)	Number of filters (d)
1	16
[HTML]9ECFD7 2	32
3	64
[HTML]9ECFD7 4	96
5	128
[HTML]9ECFD7 6	192

Table 2.1: PWC-Net filters per pyramid level

2.3.4 PWC-Net

Figure 2.8 illustrates the architecture of PWC-Net for two feature pyramid levels. To extract features, PWC-Net takes as input a RGB image pair and applies the same set of filters to each. A different amount of filters is applied to each pyramid level as shown in table 2.1, this determines the number of activation maps obtained on each pyramid level. All filters have a dimension of $d \times 3 \times 3$, where d is the amount of activation maps of each level. The filter is applied with a step of 2 in the horizontal and vertical directions, thus downsampling the input to half its resolution at each pyramid level. These set of filters correspond to the feature extraction CNNs.

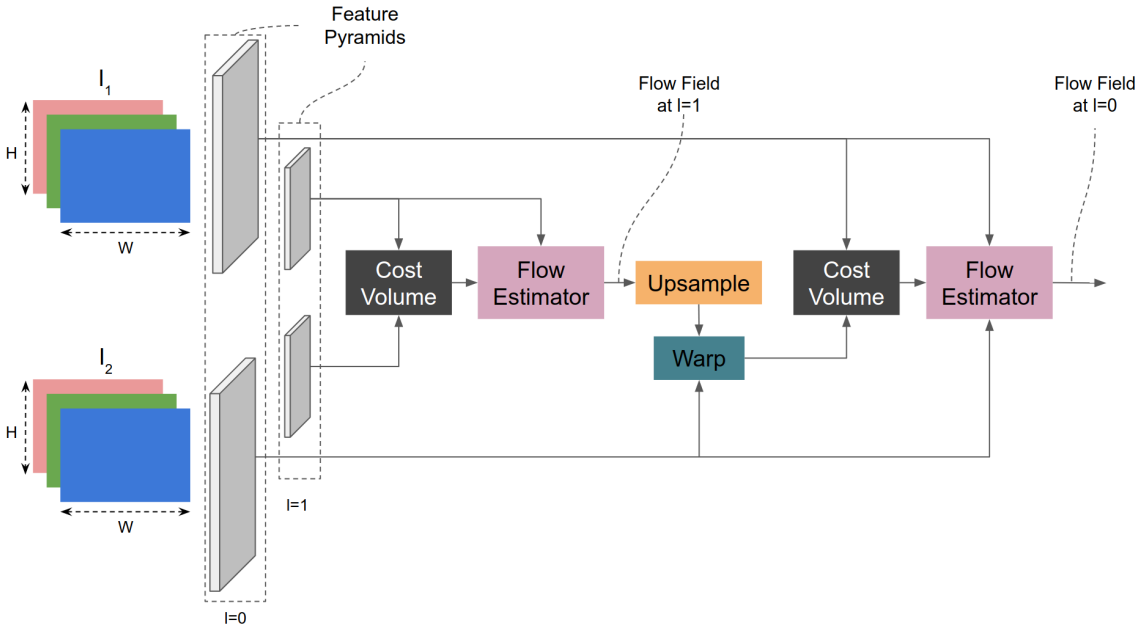

Figure 2.8: PWC-Net Architecture

For all levels, except the lower pyramid level, the previous level optical flow is up-

[HTML]409DAB Convolutional Layer	Number of feature channels
1	128
[HTML]9ECFD7 2	128
3	96
[HTML]9ECFD7 4	64
5	32

Table 2.2: PWC-Net flow estimator feature channels per flow estimator CNN layer

[HTML]409DAB Convolutional Layer	Dilation constant (k)
1	1
[HTML]9ECFD7 2	2
3	4
[HTML]9ECFD7 4	8
5	16
[HTML]9ECFD7 6	1
7	1

Table 2.3: PWC-Net context network dilation constants per level

sampled and used to warp the second image toward the first image. The warped image is then used to compute the local cost volume for that level. All the flow estimators, except for the lower level, take the level cost volume output along with the feature pyramids for that level as inputs for the flow estimator. The flow estimator is a 5 layer CNN with a fixed amount of feature channels for all levels detailed in table 2.2.

Finally, PWC-Net uses a feed-forward CNN that has 7 layers to scale the flow field to the corresponding pyramid level, known as the context network. The design of this network is based on dilated convolutions with filters of size $kx3x3$ where k is the dilation constant. Table 2.3 shows the dilation constants used by PWC-Net.

2.4 Globally Optimized Cost Volume

The cost volume represents the matching confidences between two feature maps, however, when the input images have repetitive textures with very similar appearance there can be high correlation values between multiple incorrect locations in

the cost volume [28]. Consider I_1 and I_2 images shown in figure 2.9, ideally the correlation between the green patch from I_1 and the red patch of I_2 should have the highest correlation of their feature maps, however, there are other parts of the images with very similar patterns that also show a high correlation.

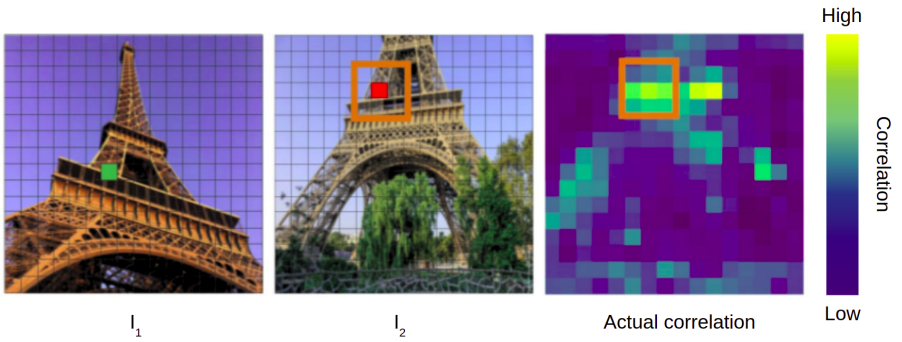

Figure 2.9: Feature correlation
Source: [28]

The ideal correlation is shown in figure 2.10, where there is only a high correlation value for I_1 green patch and it is in the location of I_2 red patch.

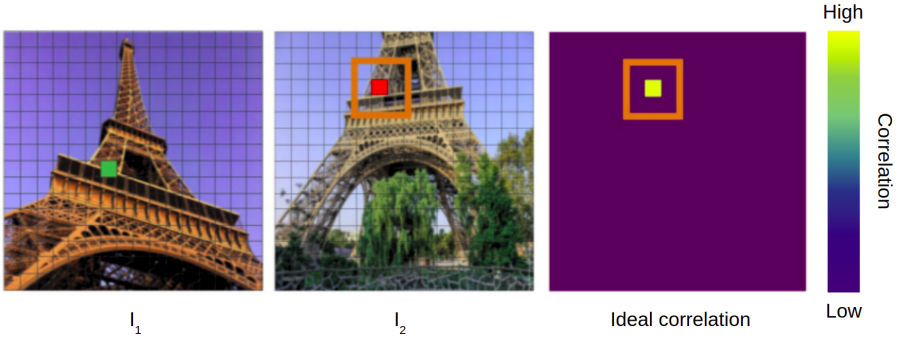

Figure 2.10: Ideal feature correlation
Source: [28]

Given that the repetitive patterns that cause the ambiguity in the correlation with I_2, usually are also in I_1, GOCor [28] proposes to use this information to generate a filtered map $w*$ that takes into account this information to improve the result of the cost volume as shown in figure 2.11.

The filtered map $w*$ is obtained according to equation 2.4. The filter P takes as input the feature maps from both images, F_1 and F_2 and adjusts the set of hyperparameters

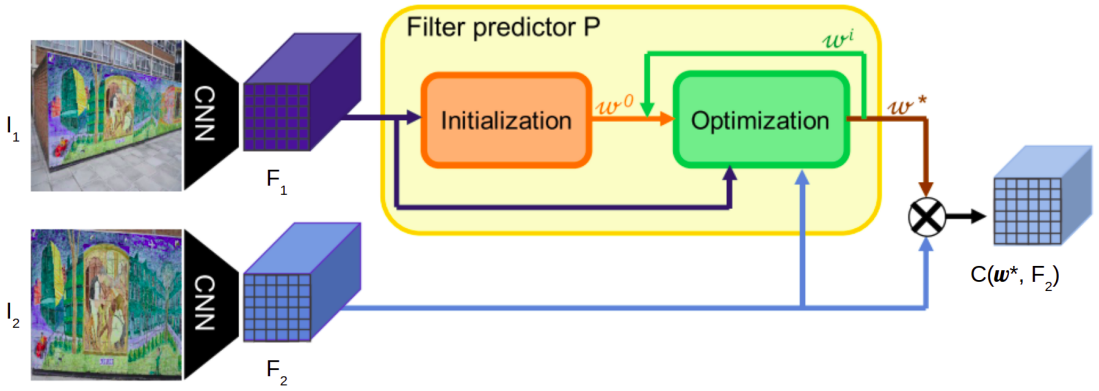

Figure 2.11: Model with GOCor
Source: [28]

θ to minimize a loss function in the forth pass of the model, this includes training and inference.

$$w^* = P_\theta(F_1, F_2) = \underset{w}{\text{argmin}} L(w; F_1, F_2, \theta) \tag{2.4}$$

The loss function has two components, each associated to an input feature map plus a regularization prior term as shown in equation 2.5, where λ_θ is a learnable weight.

$$L(w; F_1, F_2, \theta) = L_1(w; F_1, \theta) + L_2(w; F_2, \theta) + \|\lambda_\theta w\|^2 \tag{2.5}$$

The loss for the first feature map is given by 2.6 and 2.7, where y_θ is a learnable target confidence that uses the euclidean distance between two feature map regions, i, j that corresponds to the the map value that is being correlated $F_1(i, j)$ and k, l any location in F_1. v^+ and v^- are penalization weights for positive and negative correlation values. $\eta = 0$ and $\eta = 0.1$ are used in [28].

$$L_1(w; F_1, \theta) = \|\sigma_\eta(C(w, F_1); v^+, v^-) - y_\theta(\sqrt{(i-k)^2 + (j-l)^2})\|^2 \tag{2.6}$$

$$\sigma_\eta(c; v^+, v^-) = \frac{v^+ - v^-}{2}(\sqrt{c^2 + \eta^2} - \eta) + \frac{v^+ + v^-}{2}c \tag{2.7}$$

For the second feature map loss, a convolution is performed between a learnable 4D kernel R_θ and the correlation between the second feature map F_2 and the model w.

$$L_2(w; F_2, \theta) = \|R_\theta * C(w, F_2)\|^2 \tag{2.8}$$

All GOCor hyperparameter learning is based on stochastic gradient descend based on the network training loss. GOCor has been tested with supervised PWC-Net and showed a substancial enhancement for the Sintel dataset on regions with pixel velocities of more than 40 pixels, with a reduction of 1.2 pixels in the endpoint error.

2.5 Unsupervised Learning of Optical Flow Estimation

Unsupervised learning models are trained without ground truth data [1], instead a proxy objective function must be defined to optimize the CNNs hyperparameters. In 2020, Jonschkowski *et al.* presented UFlow [8], a PWC-Net based unsupervised learning framework that analyzed a set of components to determine which are key for unsupervised optical flow performance. The key components considered on the objective function for UFlow are occlusion aware photometric consistence, smoothness and self-supervision. The following subsections present these components in more detail and offers the architectural details of UFlow.

2.5.1 Occlusion Estimation

Consider figure 2.12, there are some pixels on I_1 that do not appear on I_2, such as the blue shirt cyclist that is hidden by the red shirt cyclist. For these pixels there

are no correspondences that can be established between the images, this is called occlusion [8].

Figure 2.12: Large displacements, occluded regions and texture repetition in consecutive frames
Source: Video by Kim Dae Jeung from https://pixabay.com

2.5.2 Photometric Loss

The photometric loss is the measurement of the difference between the first image and the second image warped toward the first according to the estimated optical flow [32] and this difference must be minimized to ensure photometric consistency.

2.5.3 Smoothness Loss

In addition to brightness constancy, Horn and Schunck [5] introduced the smoothness constraint. Smoothness enforces that neighboring pixels move similarly, thus have similar flow vectors. Then the smoothness loss is a measurement of the difference of the flow predictions for neighboring pixels [32].

2.5.4 Self-supervision

The optical flow estimated by a model can be used as ground truth to refine the estimation for a more difficult pair of images created by adding noise to the original image pair, as in SelFlow [13].

2.5.5 UFlow

UFlow uses a shrinked version of PWC-Net with 5 pyramid levels and 32 channels in all levels, this is to compensate for the additional memory requirements of unsupervised learning due to bi-directional losses, occlusion estimation and self-supervision [8]. In addition, UFlow randomly drops the flow estimation and pass the upscaled flow directly to the next level, this is called level dropping represented in figure 2.13 with the purple block connected to the current level flow estimator output and to the previous level upsampled flow with the green dashed arrow. UFlow also adds a residual connection from the previous level flow estimator output to the current level flow estimator shown with a red dashed arrow in figure 2.13.

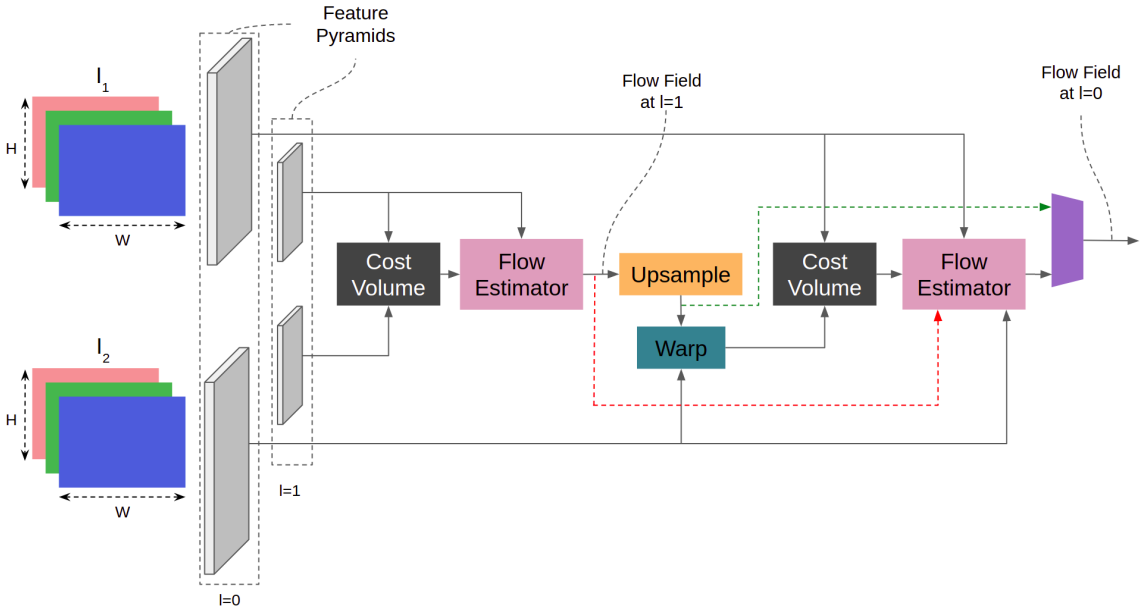

Figure 2.13: UFlow warping, cost volume and flow block

Using census loss for the photometric loss computation, UFlow reached FlowNet2 [6] supervised net performance for the KITTI 2015 dataset. For the Sintel dataset, the census loss also showed better results. For occlusion masking, the forward-backward approach showed the best results. Cost volume is computed locally in a $9x9$ neighborhood. In supervised learning of optical flow studies, the loss is typically computed until the lower pyramid level. For unsupervised learning models

for optical flow estimation, this leads to problems in the higher levels parameter learning, specifically vanishing feature activations, causing very small values on the cost volume. To address this, the cost volume is normalized using the sample mean μ_i and standard deviation σ_i of the feature maps at each level as in equation 2.9.

$$C(i,j,k) = \frac{(F_1(i,j) - \mu_1)^T}{\sigma_1} \frac{F_2(m,n) - \mu_2}{\sigma_2} \tag{2.9}$$

2.6 Related Work

The advantages offered by unsupervised models for optical flow estimation, as opposed to supervised models, have fueled a surge in studies published in this field during the recent years. In this section, we summarize the publications within our knowledge that are related to our investigation.

In UPFlow [14], the authors enhance unsupervised optical flow learning by improving feature extraction. Our work diverges in that we explore modifications for the feature correlation layer, with no adjustments made to the feature extraction network.

The research conducted by Zhang et al. [34] divides the optical estimation in two 1-D separate problems consequently separating the cost volume in one volume for the horizontal estimation and other for vertical estimation. In contrast, our study maintains the architecture of the 2D flow estimator without modification and concentrates solely on enhancements to the cost volume.

Stone et al. [22] utilize a RAFT-based [27] model architecture adapted for unsupervised learning and leverages on multi-frame flow refinement to improve the optical flow estimates among other ideas. The work of Liu et al. [12] employs a data distillation approach to unsupervised learning of optical flow. In [7], occlusion consistency is enforced using self-supervision. Notably, all three of these works utilize the traditional cost volume layer, which stands in contrast to the approach employed in our

research.

GMFlow [31] leverages a transformer network to augment the extracted features and entirely substitutes the local correlation with global matching. While their approach addresses a similar problem to our work, it employs a supervised learning model, presenting a distinction from our methodology.

Chapter 3

Hypothesis and Objectives

3.0.1 Hypothesis

The optical flow estimation for large displacements can be improved using global cost volume instead of local cost volume. However, as the global cost volume increases the computational requirements, a trade-off is to use global cost volume in the higher levels of the feature map and local cost volume in the lower levels. In addition, repetitive patterns are a problem for local and global cost volume layers, and for both cases GOCor can be used to improve the confidences of the cost volume output. Based on these ideas, our hypothesis is:

The optical flow estimation endpoint error for large displacements (40 pixels or more), with the UFlow model, can be reduced by at least 1.2 pixels[a] *using global cost volume on the two higher layers of the feature pyramids, and GOCor on all pyramid levels.*

[a] Based in the results of [28]

3.0.2 Objectives

General Objective

Reduce the large displacements average endpoint error of the optical flow estimation using a model that integrates GOCor cost volume and global cost volume.

Specific Objectives

1. Determine the additional computational load required to compute the cost volume globally in the two higher levels of the feature pyramids for UFlow and the endpoint error improvement for large displacements.
2. Integrate GOCor into the UFlow model and evaluate the improvement to the optical flow estimation for large displacements.
3. Evaluate the optical flow estimation endpoint error for large displacements using both GOCor and global cost volume for the two higher levels of the feature pyramids of UFlow.

Chapter 4

Methodology

This chapter outlines the methodology employed in the experiments conducted and describes the metrics utilized to validate the hypothesis.

4.1 Model

The unsupervised learning model selected for the experiments is UFlow, incorporating the components that demonstrated the most favorable results for the Sintel dataset evaluation in [8]. In addition, we employ the GOCor module as presented in Truong et al. [28] to enhance the cost volume block of the UFlow model.

4.2 Dataset

We use 1593 RGB images of size 1024×448 from the Sintel Final dataset for training and evaluation with the default train-test data split. The Sintel Final dataset provides 1041 images for training with ground truth flow and 552 images for evaluation without ground truth flow. We selected the Sintel Final dataset for training and evaluation instead of the Sintel Clean dataset because the Final pass includes blur and atmospheric effects, making it more challenging than the Clean pass[2] and a

closer approximation to real world images.

4.3 Training

The training of the UFlow model does not rely on the ground truth flow, thus we conduct the training on the test split of the dataset and assess performance on the training split, following a practice commonly observed in other studies. UFlow is trained by minimizing a loss function composed by three terms: the photometric loss, the smoothness regularization loss and the self-supervision loss as shown in equation 4.1 respectively.

$$\mathcal{L}(D, \theta) = w_{photo}\mathcal{L}_{photo} + w_{smooth}\mathcal{L}_{smooth} + w_{self}\mathcal{L}_{self} \tag{4.1}$$

For the photometric loss term, a constant weight of $w_{photo} = 1$ is used throughout all the training process. For the smoothness regularization term, first order smoothness is used with a constant weight of 2 and a constant smoothness edge-weight of 150. Self-supervision is disabled for the initial half of the training process ($w_{self} = 0$) and it is incremented linearly to 0.3 over the subsequent 10% of the training steps.

In addition to the parameters described above, UFlow implementation offers several other parameters that can be adjusted to fine-tune the training strategy. Table 4.1 provides a description of the parameters deemed relevant for this work.

Due to the memory constraints of the GPUs employed in this work, we use batch size 1 consistently throughout all experiments and scale down the input images resolution from 1024×448 to 512×224.

[HTML]409DAB Parameter	Description
num_train_steps	The number of iterations or updates made to the model during the training stage.
batch_size	The number of images to process before updating the model
geometric_augmentation	Whether to augment the training dataset with vertical/horizontal flips.
selfsup_after_num_steps	Whether to enable the self supervision feature after a number of training steps.
selfsup_ramp_up_steps	The number of training steps in which the self supervision weight is increased linearly by 0.3.
weight_selfsup	The initial weight for self supervision.
weight_photo	The weight for the photometric loss.
weight_smooth1	The weight for the first order smoothness loss.
weight_smooth2	The weight for the second order smoothness loss.
lambda	The edge-weight for the smoothness
gpu_learning_rate	The learning rate.
lr_decay_after_num_steps	The number of training steps after which the learning rate value decays.
lr_decay_steps	The number of training steps in which the learning rate is decayed.
lr_decay_type	The function to be used for learning rate decay.

Table 4.1: UFlow Training Parameters

In [8], the UFlow training with batch size of 1 was conducted over 1200 epochs, ech epoch consisting of 1000 training steps. In our work, we opted to reduce the training steps to 120 epochs in the initial trials and further down to 50 epochs in the later experiments. This adjustment was made to align with the computational capabilities of the machines utilized in our research, as training for 1200 epochs would have been impractical in our case due to the considerable time it takes on the machines employed in this work.

In alignment with the techniques that yielded the most favorable results in the work of Jonschkowski et al. [8], we enable augmentations to the training set: random swapping of color channels, hue shifting and vertical/horizontal flipping and the learning rate (how much weights change on each iteration of backpropagation) is set

initially to 1^{-4} and decayed exponentially after $\frac{5}{6}$ of the total training steps.

Table 4.2 presents the training parameters values used in the experiments trained for 120 epochs and Table 4.3 shows the values used in the experiments trained for 50 epochs.

It is a common practice to pre train the network on a different dataset before the training with Sintel, as in [28] and [8], in this study, however, pre-training is omitted with the aim of mitigating the computational demands required for each experiment.

[HTML]409DAB Parameter	Value
num_train_steps	120 k
batch_size	1
geometric_augmentation	True
selfsup_after_num_steps	60 k
selfsup_ramp_up_steps	12 k
weight_selfsup	0.3
weight_photo	1.0
weight_smooth1	4.0
weight_smooth2	0
lambda	150
gpu_learning_rate	0.0001
lr_decay_after_num_steps	100 k
lr_decay_steps	20 k
lr_decay_type	exponential

Table 4.2: UFlow Baseline parameters for training on Sintel Test Set for 120 epochs

4.4 Experiments

The experiments outlined in this research focus on the cost volume block within the UFlow model. Our primary objective is to enhance the optical flow estimation specifically for scenarios involving large displacements.

We evaluate three factors: the model architecture, the search neighborhood for the cost volume and the number of training epochs.

[HTML]409DAB Parameter	Value
num_train_steps	50 k
batch_size	1
geometric_augmentation	True
selfsup_after_num_steps	25 k
selfsup_ramp_up_steps	5 k
weight_selfsup	0.3
weight_photo	1.0
weight_smooth1	4.0
weight_smooth2	0
lambda	150
gpu_learning_rate	0.0001
lr_decay_after_num_steps	42 k
lr_decay_steps	8 k
lr_decay_type	exponential

Table 4.3: UFlow Baseline parameters for training on Sintel Test Set for 50 epochs

4.4.1 Model Architecture

Two levels of of model architecture are employed: the original architecture of UFlow published in [8] and the UFlow augmented with the GoCor published in [28].

4.4.2 Cost Volume Search Neighborhood

Two levels are used, local cost volume with a search neighborhood of 4 pixels and global cost volume.

4.4.3 Number of Training Epochs

The initial experiments in this investigation endured training for 120 epochs. However, due to varying computational demands, some experiments were deemed impractical to train for the full 120 epochs with the hardware utilized in our work. Consequently, all experiments were executed for 50 epochs instead, including the experiments that were trained for 120 epochs initially.

Tables 4.4 and 4.5 show the factor-levels possible combinations. We conducted six experiments, the **Baseline-120E** and **Baseline-50E** experiments provide the baseline results that we use to evaluate the improvements suggested in this research.

Experiments **GCV-2Top-120E**, **GCV-2Top-50E** and **GoCor-All-50E** aim to assess the impact of modifying one of the factors, either model architecture or search neighborhood of the cost volume, to a level different than the employed in baseline. Experiment **GCV-2Top-GoCor-All-50E** assesses the combined effects of altering both factors from the baseline.

[HTML]409DAB [HTML]FFFFFF	Model Architecture	
[HTML]939599 Search Neighborhood	[HTML]81d0db UFlow	[HTML]81d0db UFlow + GOC
[HTML]c7cad1 Local S=4	Baseline-120E	Not executed
[HTML]c7cad1 Global	GCV-2Top-120E	Not executed

Table 4.4: Experiments trained for 120 epochs

[HTML]409DAB [HTML]FFFFFF	Model Architecture	
[HTML]939599 Search Neighborhood	[HTML]81d0db UFlow	[HTML]81d0db UFlow + GOC
[HTML]c7cad1 Local S=4	Baseline-50E	GOCor-All-50E
[HTML]c7cad1 Global	GCV-2Top-50E	GCV-2Top-GOCor-All-50E

Table 4.5: Experiments trained for 50 epochs

4.5 Limitations and Scope

4.5.1 Evaluation

The main objective of this work is aligned with reducing the average endpoint errors in optical flow estimations, particularly for large displacements. For the purpose of this study, we define large displacements as those with a ground truth optical flow surpassing 40 pixels and evaluate the research hypothesis using this criteria. In addition to assessing the endpoint error for large displacements, we gather additional evaluation metrics to comprehensively analyze the broader impacts of the proposed changes. These metrics are detailed in Table 4.6.

[HTML]409DAB Metric	Description
EPE (pixels)	Endpoint error, the difference between the predicted flow and the ground truth flow, computed with the formula $\sqrt{(u_0 - u_1)^2 + (v_0 - v_1)^2}$, where (u_0, v_0) is the predicted flow and (u_1, v_1) is the ground truth flow, averaged over all the frames.
s0-10 (pixels)	Endpoint error of the pixels with ground truth velocities less than 10 pixels, averaged over all the frames.
s10-40 (pixels)	Endpoint error of the pixels with ground truth velocities between 10 and 40 pixels, averaged over all the frames.
s40+ (pixels)	Endpoint error of the pixels with ground truth velocities greater than 40 pixels, averaged over all the frames.
Average Training Step Time (ms)	The training step time averaged over all the steps.
Average Inference Time (ms)	The optical flow inference time averaged over all inferences for the images in the training dataset.

Table 4.6: Evaluation Metrics

The dataset selected for the evaluation is the Sintel Final training split. The training split is used for evaluation because it provides ground truth that is used to compute the estimation errors. Other datasets are available, however, we chose Sintel Final because it comprises realistic effects such as motion blur, atmospheric effects and large displacements (45 M pixels with velocity of more than 40 pixels between consecutive frames throughout the 1041 images of the training split).

The endpoint error ground truth values are provided at the MPI Sintel Dataset website. Table 4.5.1 shows the percentage of optical flow ground truth data points available for each velocity category. The large displacements account for almost 10% of the motion.

[HTML]409DAB heightCategory	Ground truth data points
s0-10	69%
s10-40	21.27%
s40+	9.73%

Table 4.7: Percentage of data within each displacement category in the MPI Sintel Dataset

4.5.2 Computational Resources

We observed that the training process imposes a substantial demand on GPU resources; rendering it impractical for execution in a personal laptop. Consequently, we conducted both training and inference on servers equipped with powerful NVIDIA GPUs. In our study, we used two machines with the specifications detailed in Table 4.8.

[HTML]409DAB heightSpecification	Kabré Nukwa Server	Google Cloud Compute Instance
GPU	NVIDIA Tesla K40c	NVIDIA L4
GPU Memory	12 GB	24 GB
CPU	Intel Xeon (4 cores)	Intel Cascade Lake (4 vCPUs)
RAM	16 GB	16 GB
GPU compute capability	3.5	8.9

Table 4.8: Technical specifications of the servers used

Kabré Nukwa is a high-performance computing cluster facilitated at no cost by the National Center for High Technology of Costa Rica (CeNAT) to Technological Institute of Costa Rica students engaged in research. Users are subject to certain limitations, including a maximum allowable time between 72 and 168 hours per computational node and a cap of two nodes per person. Upon reaching the maximum allocated time, the training process is halted, and the task is queued to resume execution when the node becomes available again—once other users in the queue have completed their allotted time slices.

We faced a limitation in the execution of GOCor on the Kabré Nukwa server: the compute capability of the GPU is incompatible with the installation of the pre-built software packages utilized by the GOCor implementation, thus we opt for using Google Cloud Compute for the GOCor experiments.

We discovered that the GPU memory requirements for experiments incorporating GOCor and global cost volume simultaneously, exceed the capacity of the Kabré server. Consequently, we chose to utilize Google Cloud Compute for the training

and inference of these cases. The Google Cloud Compute instances used incurred a cost of $17.7 for each day of usage.

Both servers employed have a Linux based OS, and thus we use the *top* and *nvidia-smi* commands to assess the CPU load, RAM usage, GPU utilization and GPU memory usage.

To manage the cost associated with training the experiments on Google Cloud Compute within budget, and the time required to train in Kabré within a reasonable timeline, we impose the following constraints:

1. We use half image resolution in all our experiments (224x512).
2. We train experiments on the Kabré server for a maximum of 120 epochs, and for experiments executed on the Google Cloud Compute instance, we limit the training to a maximum of 50 epochs.
3. We skip pre-training.

Chapter 5

Results and Analysis

5.1 Baseline

For the baseline experiments, the original UFlow model is used without changes to the cost volume block. The results presented in this section serve as the reference points for evaluating the efficacy of the improvements proposed in this work.

5.1.1 Baseline-120E and Baseline-50E

Tables 4.2 and 4.3 summarize the parameters used for this experiment. For Baseline-120E, training was executed for 120 epochs each of 1000 steps on the Sintel final dataset testing split, and for 50 epochs for each of the two Baseline-50E replicates. Tables 5.1 and 5.2 show the training results.

Kabré Nukwa uses 100% of CPU load, and allocates 11.6GB of GPU memory for both training and evaluation. The training demands almost 5GB of RAM, while the evaluation uses around 2GB of RAM. The utilization of the GPU for the training is of 70% and of 32% for the evaluation.

In the Google Cloud Compute case, the CPU usage behaves similarly in both training and evaluation, with up to 126% of load. 22GB of GPU memory are allocated for

both training and evaluation. The GPU utilization is of 14% for training and 3% for evaluation. The RAM usage is of around 6GB for training and around 3GB for evaluation.

[HTML]409DAB height Metric	Result
Server	Kabré Nukwa
Total Training Steps	120000
Total Training Time (h)	95
Average Step Time (ms)	2848
Final Epoch Total Loss	2.35

Table 5.1: Training results for baseline experiment with 120 epochs

[HTML]409DAB height Metric	1^{st} Replicate Result	2^{nd} Replicate Result
Server	Google Cloud Compute	Kabré Nukwa
Total Training Steps	50000	50000
Total Training Time (h)	64	42
Average Step Time (ms)	4609	2925
Final Epoch Total Loss	2.41	2.4

Table 5.2: Training results for baseline experiments with 50 epochs

Figures 5.1, 5.2, 5.3 and 5.4 show the values of the losses in equation 4.1 as the training epochs increase. The photometric (Figure 5.1) and smoothness (Figure 5.2) losses show the steepest decrease during the first 40 epochs of the training. The self-supervision loss (Figure 5.3) is zero in the first 60 epochs because self-supervision is enabled only in the last 60 epochs of the training, where it shows a small decrease from 0.05 to 0.045.

Tables 5.3 and 5.4 show the evaluation results.

[HTML]409DAB height Metric	Result
EPE (pixels)	5.26
s0-10 (pixels)	2.38
s10-40 (pixels)	6.67
s40+ (pixels)	38.10
Average Inference Time (ms)	863

Table 5.3: Inference results for baseline experiment with 120 epochs

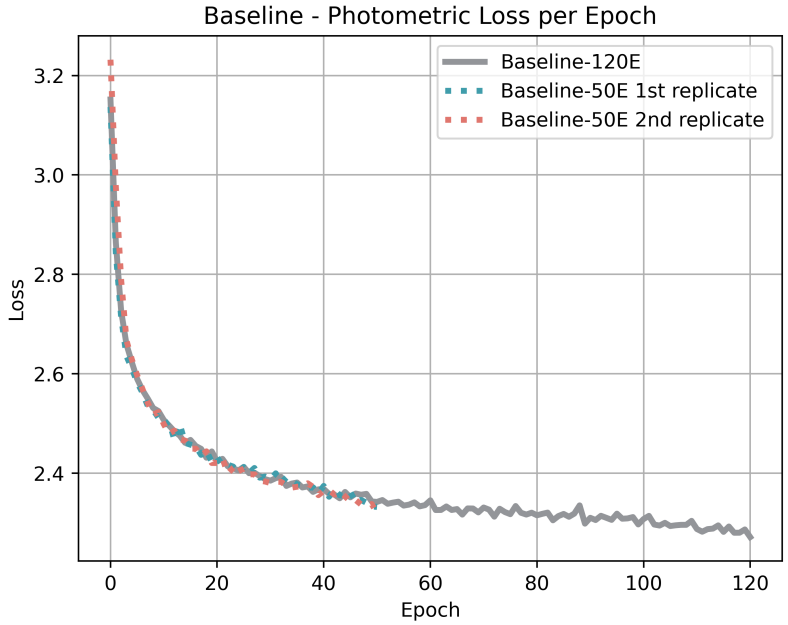

Figure 5.1: Baseline Training Photometric Loss per Epoch

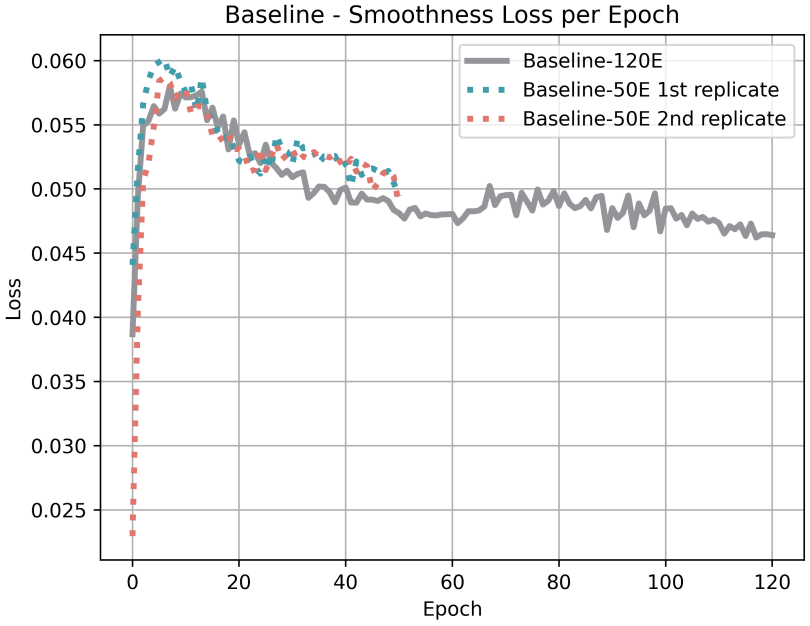

Figure 5.2: Baseline Training Smoothness Loss per Epoch

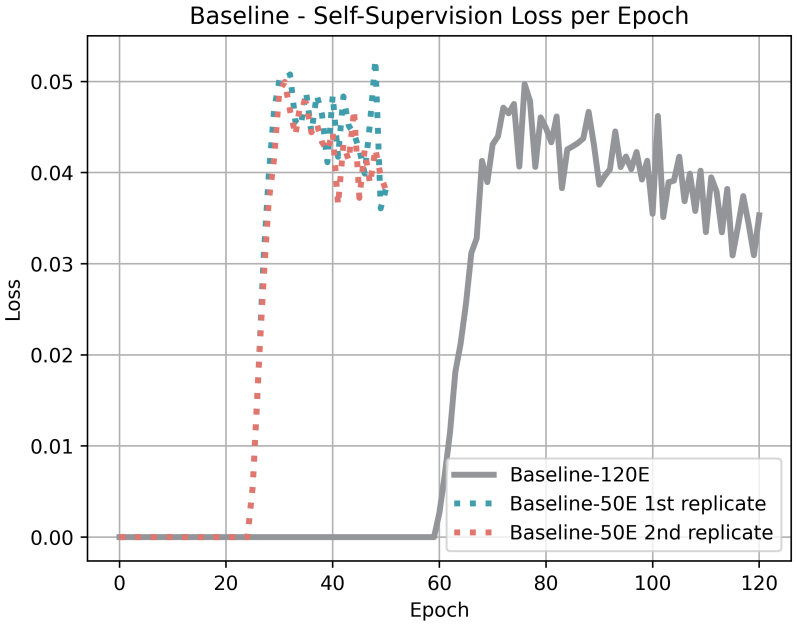

Figure 5.3: Baseline Training Self-Supervision Loss per Epoch

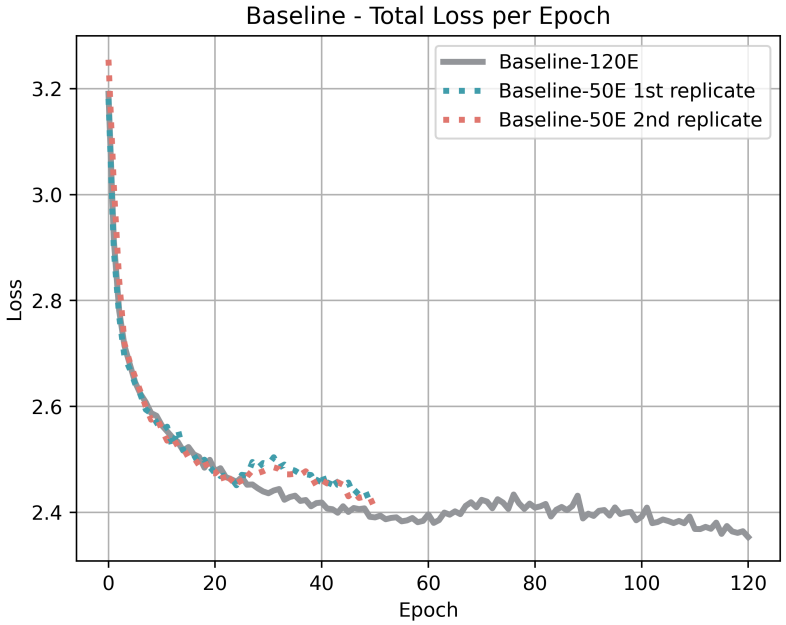

Figure 5.4: Baseline Training Total Loss per Epoch

[HTML]409DAB height Metric	1^{st} Replicate Result	2^{nd} Replicate Result
EPE (pixels)	5.78	5.98
s0-10 (pixels)	2.54	2.66
s10-40 (pixels)	7.03	7.26
s40+ (pixels)	41.30	41.4
Average Inference Time (ms)	728	445

Table 5.4: Inference results for baseline experiments with 50 epochs

5.2 Global Cost Volume

The implementation of global cost volume on the two highest pyramid levels ($\ell = 3$ and $\ell = 4$) changes the dimensions of the cost volume tensor. For local cost volume, the size of the output tensor is described in equation 5.1:

$$C^{\ell} \in \mathbb{R}^{\frac{W}{\ell} \times \frac{H}{\ell} \times S^2} \tag{5.1}$$

The baseline UFlow uses local cost volume with a search neighborhood $S = 9$, therefore, the local cost volume tensor size for all pyramid levels is $C^{\ell} \in \mathbb{R}^{\frac{W}{\ell} \times \frac{H}{\ell} \times 81}$. When the search neighborhood is expanded to the full image, then the global cost volume tensor size is given by equation 5.2:

$$C^{\ell} \in \mathbb{R}^{\frac{W}{\ell} \times \frac{H}{\ell} \times (\frac{W}{\ell} \times \frac{H}{\ell})} \tag{5.2}$$

Table 5.5 shows the pyramid levels resolutions and the cost volume dimensions for UFlow with global cost volume in the two highest pyramid levels. UFlow uses a pyramid with the base level resolution that is a half of the input resolution and computes the cost volume only for pyramid levels with $\ell > 0$.

5.2.1 GCV-2Top-120E and GCV-2Top-50E

Tables 5.6, 5.7, 5.8 and 5.9 present the training an inference results for the global cost volume experiments respectively. The experiment with 50 epochs exhibits a

[HTML]409DAB Pyramid level (ℓ)	Number of filters (d)	Width	Height	Cost volume size
0	32	512	224	-
[HTML]9ECFD7 1	32	256	112	$256 \times 112 \times 81$
2	32	128	56	$128 \times 56 \times 81$
[HTML]9ECFD7 3	32	64	28	$64 \times 28 \times 1792$
4	32	32	14	$32 \times 14 \times 448$

Table 5.5: Cost volume size per pyramid level

similar trend than the experiment with 120 epochs, as illustrated in Figures 5.5, 5.6, 5.7 and 5.8. The training process renders a CPU load of around 100%, a RAM usage around 5GB, up to 75% of GPU utilization and reaches up to 11.63 GB of GPU memory in the Kabré Nukwa server. The inference shows similar requirements in all aspects but RAM, as it utilizes around 1.8GB.

[HTML]409DAB height Metric	Result
Server	Kabré Nukwa
Total Training Steps	120000
Total Training Time (h)	135
Average Step Time (ms)	3996
Final Epoch Total Loss	2.36

Table 5.6: Training results for the GCV experiment with 120 epochs

[HTML]409DAB height Metric	1^{st} Replicate Result	2^{nd} Replicate Result
Server	Kabré Nukwa	Kabré Nukwa
Total Training Steps	50000	50000
Total Training Time (h)	66	82
Average Step Time (ms)	4556	4310
Final Epoch Total Loss	2.42	2.43

Table 5.7: Training results for the GCV experiments with 50 epochs

Tables 5.8 and 5.9 shows the evaluation results.

5.3 Local GOCor

In this section, we present the outcomes of integrating the GOCor module into all the correlation levels.

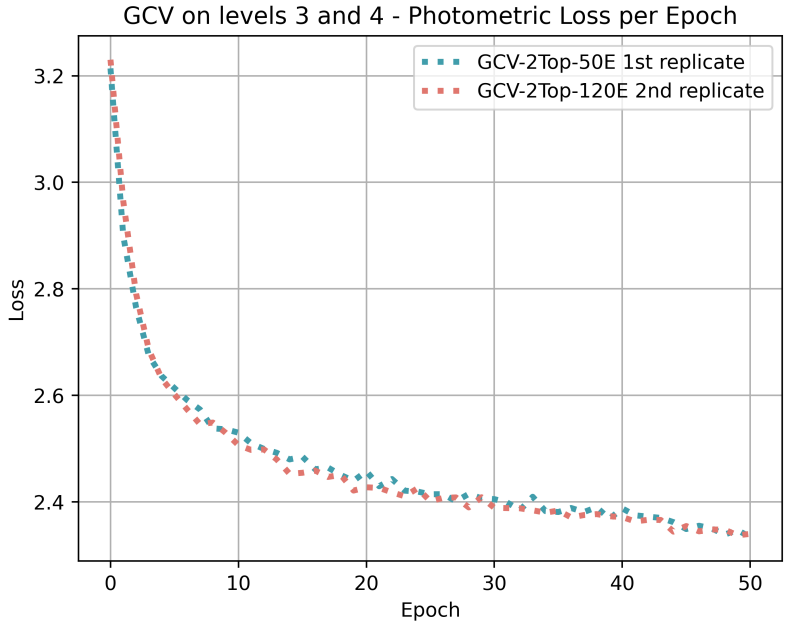

Figure 5.5: GCV Training Photometric Loss per Epoch

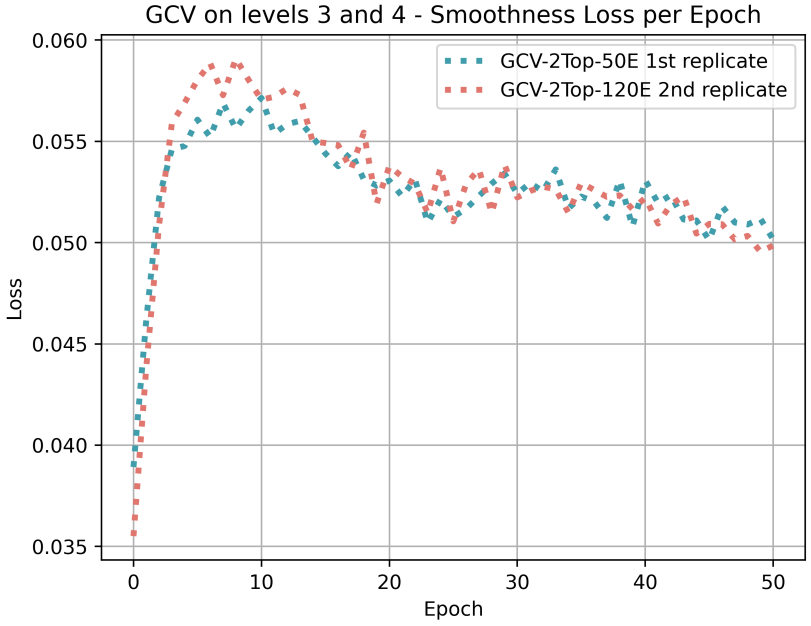

Figure 5.6: GCV Training Smoothness Loss per Epoch

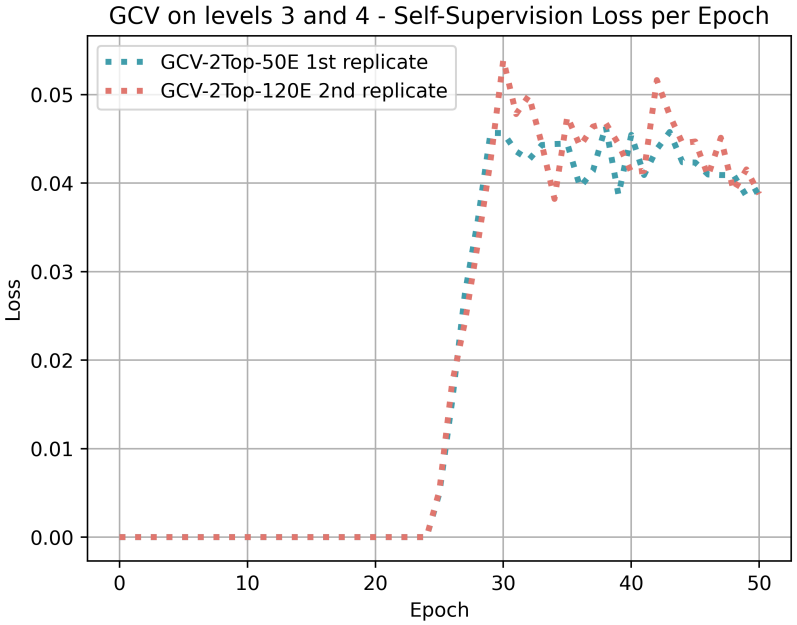

Figure 5.7: GCV Training Self-Supervision Loss per Epoch

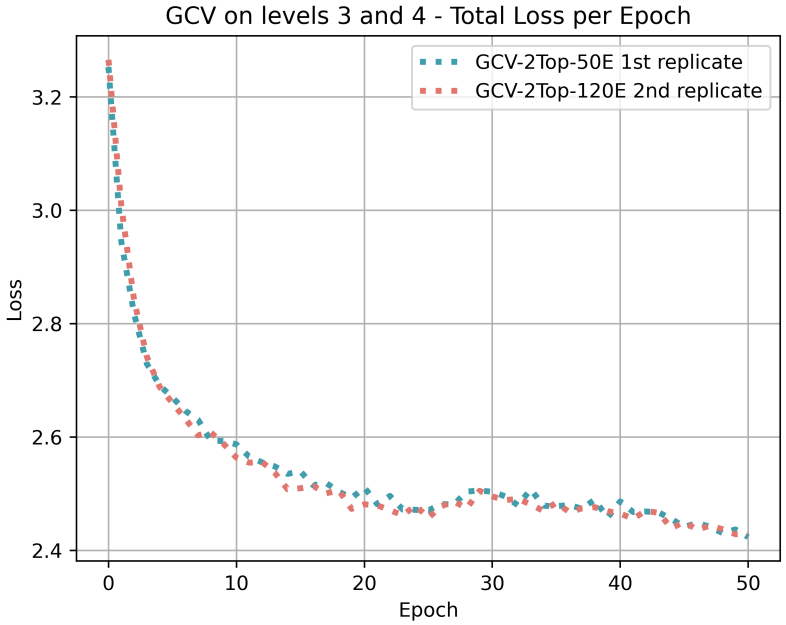

Figure 5.8: GCV Training Total Loss per Epoch

[HTML]409DAB height Metric	Result
EPE (pixels)	5.59
s0-10 (pixels)	2.39
s10-40 (pixels)	6.61
s40+ (pixels)	40.38
Average Inference Time (ms)	905

Table 5.8: Evaluation results for the GCV experiment with 120 epochs

[HTML]409DAB height Metric	1^{st} Replicate Result	2^{nd} Replicate Result
EPE (pixels)	5.91	5.8
s0-10 (pixels)	2.49	2.61
s10-40 (pixels)	7.01	7
s40+ (pixels)	41.63	40.76
Average Inference Time (ms)	1000	614

Table 5.9: Evaluation results for the GCV experiments with 50 epochs

5.3.1 GOCor-All-50E

The results for this experiment are presented in Tables 5.10 and 5.11. Figures 5.9, 5.10, 5.11 and 5.12 depict the training process.

For the training and inference of this experiment, we noticed a GPU utilization of up to 100%, up to 127% of CPU load, and a GPU memory load of up to 22.39GB in the Google Cloud Compute instance. Regarding the main memory usage, it reaches up to 6.5GB of RAM during training and up to 3GB during evaluation.

[HTML]409DAB heightMetric	1^{st} Replicate Result	2^{nd} Replicate Result
Server	Google Cloud Compute	Google Cloud Compute
Total Training Steps	50000	50000
Total Training Time (h)	170	70
Average Step Time (ms)	12205	5426
Final Epoch Total Loss	2.47	2.54

Table 5.10: Training results for GOCor-All-50E experiments

Table 5.11 shows evaluation results.

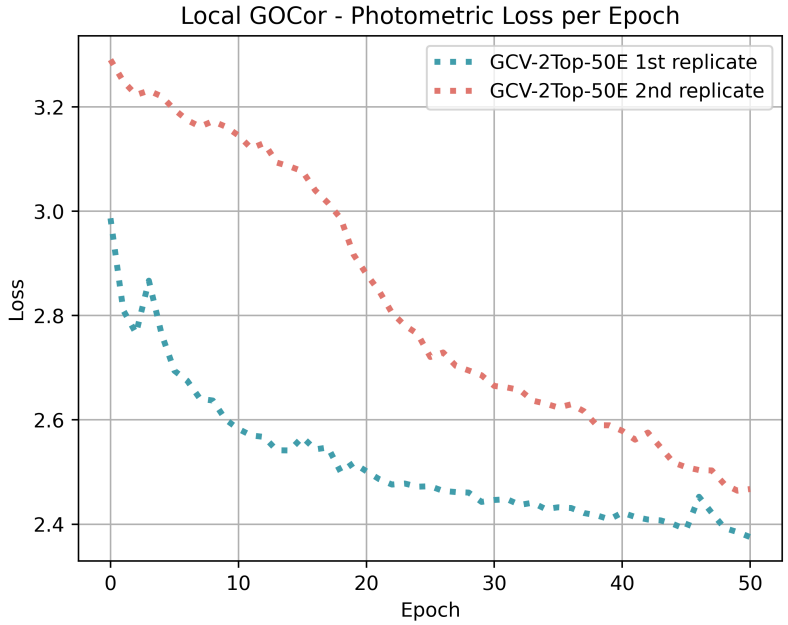

Figure 5.9: GOCor-All-50E Training Photometric Loss per Epoch

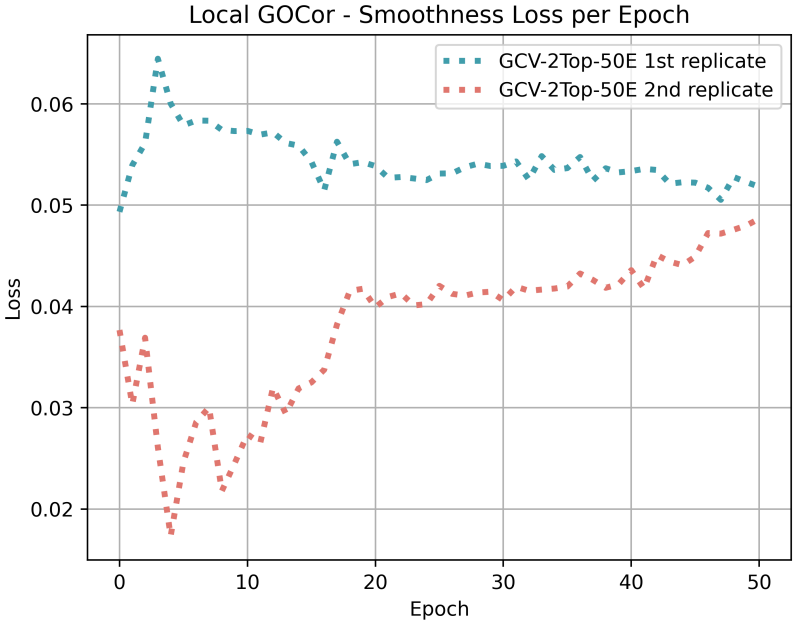

Figure 5.10: GOCor-All-50E Training Smoothness Loss per Epoch

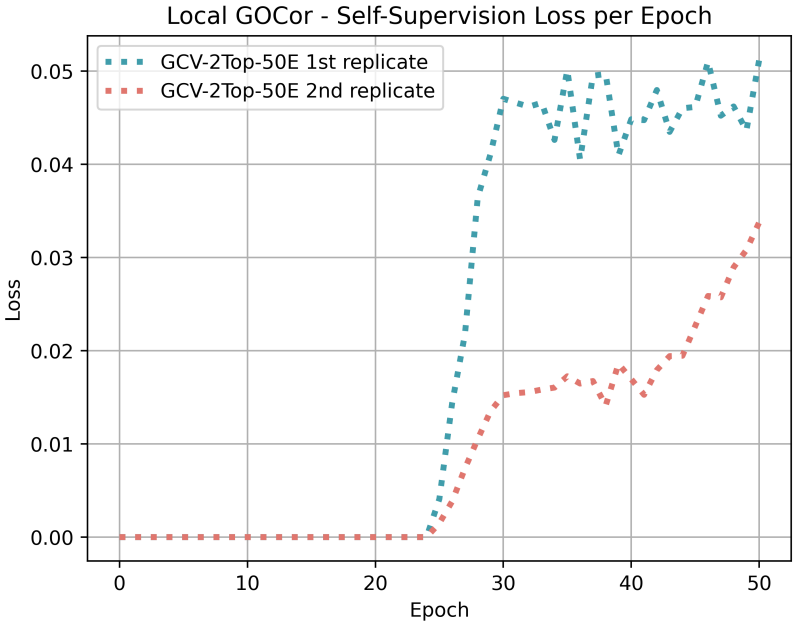

Figure 5.11: GOCor-All-50E Training Self-Supervision Loss per Epoch

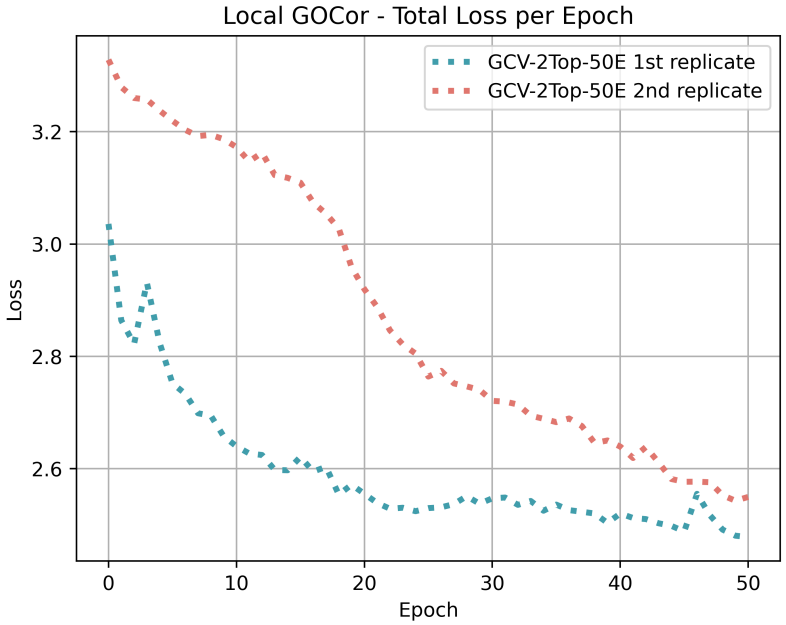

Figure 5.12: GOCor-All-50E Training Total Loss per Epoch

[HTML]409DAB heightMetric	1^{st} Replicate Result	2^{nd} Replicate Result
EPE (pixels)	5.83	
s0-10 (pixels)	3	
s10-40 (pixels)	7.74	
s40+ (pixels)	39.33	
Average Inference Time (ms)	10999	

Table 5.11: Inference results for the GOCor-All-50E experiments

5.4 Global Cost Volume and GOCor

This section presents the results of using global cost volume on the two higher levels of the feature pyramids concurrently with GOCor in all the pyramid levels.

5.4.1 GCV-2Top-GOCor-All-50E

The results for this experiment are presented in Tables 5.12 and 5.13. Figures 5.13, 5.14, 5.15 and 5.16 illustrate the evolution of the training process for these experiments.

A GPU memory of around 22GB is required for this experiment. The utilization of the GPU is of up to 100% with a CPU load of up to 126%. The maximum RAM required is around 6GB for training and around 3GB for evaluation.

[HTML]409DAB heightMetric	1^{st} Replicate Result	2^{nd} Replicate Result
Server	Google Cloud Compute	Google Cloud Compute
Total Training Steps	50000	50000
Total Training Time (h)	267	116
Average Step Time (ms)	19277	8352
Final Epoch Total Loss	2.47	2.54

Table 5.12: Training results for the GCV-2Top-GOCor-All-50E experiments

Table 5.13 shows evaluation results.

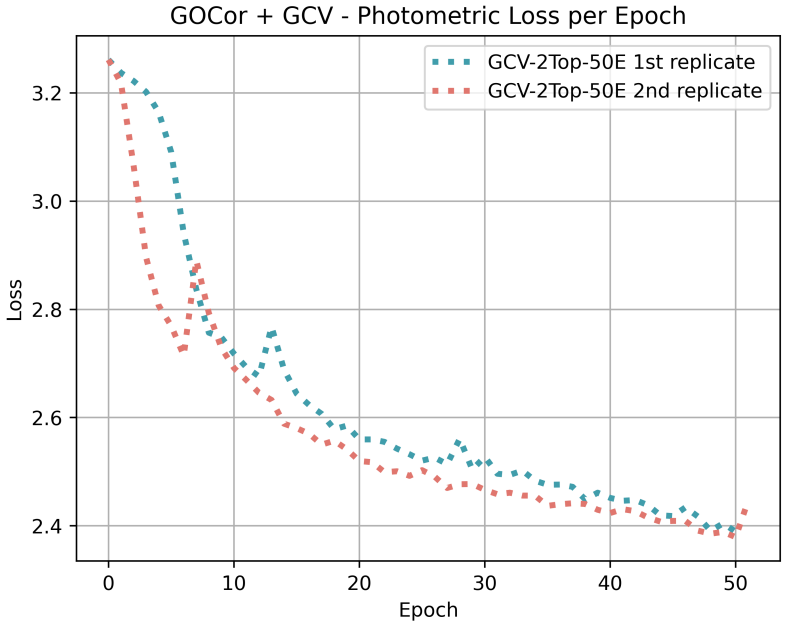

Figure 5.13: GCV-2Top-GOCor-All-50E Training Photometric Loss per Epoch

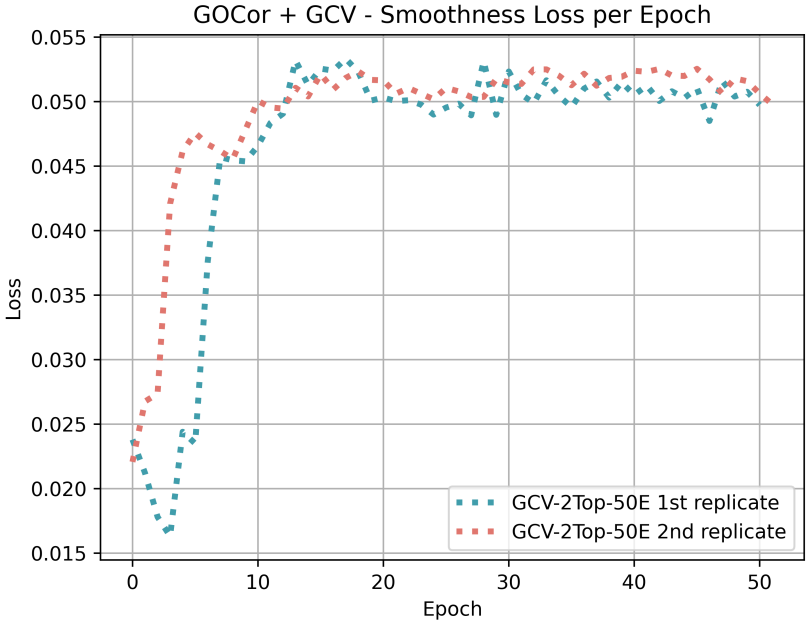

Figure 5.14: GCV-2Top-GOCor-All-50E Training Smoothness Loss per Epoch

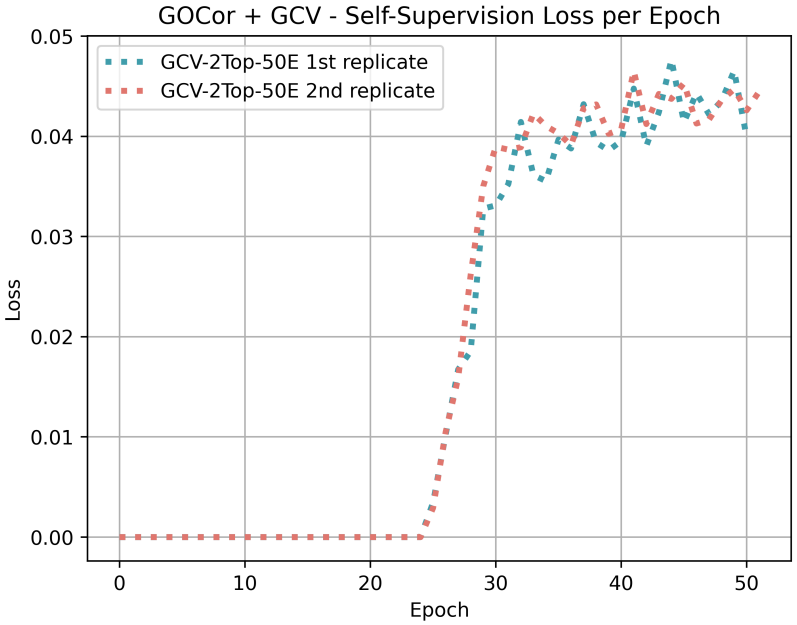

Figure 5.15: GCV-2Top-GOCor-All-50E Training Self-Supervision Loss per Epoch

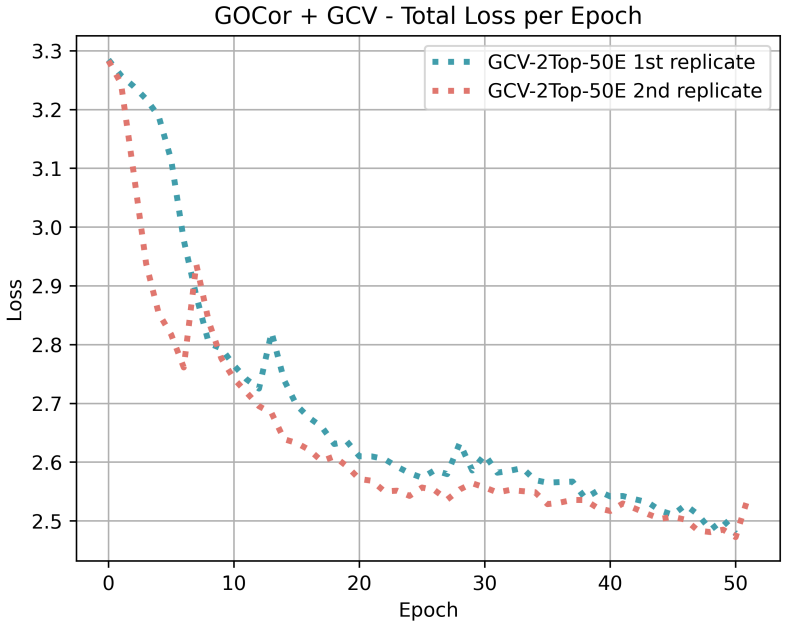

Figure 5.16: GCV-2Top-GOCor-All-50E Training Total Loss per Epoch

[HTML]409DAB heightMetric	1^{st} Replicate Result	2^{nd} Replicate Result
EPE (pixels)	6.26	6.04
s0-10 (pixels)	2.75	2.73
s10-40 (pixels)	7.4	7.28
s40+ (pixels)	41.87	41.19
Average Inference Time (ms)	1253	7154

Table 5.13: Evaluation results for the GCV-2Top-GOCor-All-50E experiments

5.5 Analysis

5.5.1 Impact of the Reduction of the Number of Training Epochs

We observed consistent patterns in Figures 5.4 and 5.8, where the total loss demonstrated a sharp decrease in the initial 20 epochs, followed by a more gradual decline. Notably, experiments trained for 50 epochs exhibited a slight increase in loss around epoch 25, coinciding with the introduction of self-supervision.

In Tables 5.3 and 5.4, the results indicate a noteworthy improvement of 0.52 pixels in average EPE and a substantial enhancement in the s40+ EPE metric by 3.2 pixels when the training steps increased from 50 to 120 epochs for the baseline experiments. However, experiments incorporating the global cost volume on the top two levels of the feature pyramid showed more modest gains—0.32 pixels for average EPE and 1.25 pixels for large displacements—compared to baseline results as shown in Tables 5.8 and 5.9.

Concerning the total training time, we observed a non-linear behavior, where certain training epochs require notably more time than others. Examining the results for global cost volume in Tables 5.6 and 5.7 as an example, both experiments were conducted on the Kabré server but on distinct nodes. Interestingly, the average step time is lower overall for the experiment with 120 epochs compared to the one with 50 epochs. In spite of the fact that the average training step time is lower in the experiment with 120 epochs, opting for only 50 epochs cut down the training time by

69 hours. In the case of the baseline experiments, although these were executed in different servers, they exhibit a similar pattern to that of the the global cost volume in terms of training time.

Despite the improvements observed with the increase in training epochs, we have opted to maintain a 50-epoch training duration for subsequent experiments. This decision answers to the limitation of keeping the training costs within constraints.

5.5.2 Impact of the global cost volume

The sole modification of replacing the local cost volume for global cost volume in the correlation of the two higher levels of the feature pyramids does not yield improvements in optical flow estimation for large displacements.

Indeed, the outcomes presented in Tables 5.4 and 5.9 reveal a deterioration in the error for large displacements, worsening by 0.33 pixels.

This finding indicates that a broader search window for the feature correlation in the feature levels leads to an increase in the number of mismatched features. This results confirms our initial intuition that the influence of the pattern repetition is exacerbated by enlarging the correlation search window and the sole use of global cost volume is not enough to improve the optical flow estimation results.

An interesting direction for future experimentation involves exploring the augmentation of the search window for correlation in the lower levels of the feature pyramids, where there is greater image resolution, as opposed to the higher levels. This exploration would aim to evaluate the effects of scaling on correlation mismatches caused by pattern repetition.

Both Baseline-120E and GCV-2Top-120E are trained on the same server. We observe that it takes around 40 hours more to train the experiment that implements global cost volume as per the results presented in Tables 5.1, 5.2, 5.6 and 5.7. In addition,

the inference time increased by 42 ms.

Both GCV-2Top and baseline trainings can be executed within the hardware capabilities of the Kabré Nukwa server.

5.5.3 Impact of GOCor

The outcomes of employing local GOCor in the correlation of all pyramid layers, as depicted in Table 5.11, reveal a significant improvement of 1.97 pixels in the optical flow estimation for large displacements in comparison with the baseline experiment. Surprisingly, the use of local GOCor led to degradation of the other endpoint error metrics; nevertheless, the overall increase in the average EPE is of merely 0.05 pixels.

5.5.4 Combined impact of global cost volume and GOCor

Contrary to the initial intuition that formed the hypothesis of this study, the substitution of the local cost volume for the global cost volume in the correlation of the two higher layers of the feature pyramids alongside GOCor does not result in improvements in optical flow estimation for large displacements. In fact, we found an increase of the endpoint error for large displacements of 0.57 with respect to the baseline results.

Given that employing GOCor alone in all levels produced improved outcomes for large displacements, we posit that utilizing an expanded search window in the two top levels of the feature pyramid may contribute to an increased number of mismatches, due to scaled down pattern repetition. These mismatches could be propagated by the model architecture to the lower levels of estimation, resulting in diminished results.

The GPU memory demanded to train the implementation of both GOCor and global cost volume increases in contrast with the baseline, GOCor only and global cost volume only experiments. We found that the concurrent use of both enhancements

require more than 12GB of GPU memory.

5.5.5 Overall Comparison

Figure 5.17 illustrates the endpoint error results obtained for all the experiments in this study. The use of local GOCor in all levels of the feature pyramids yields an improvement in the optical flow estimation for large displacements, all the other experiments show a deterioration in this metric.

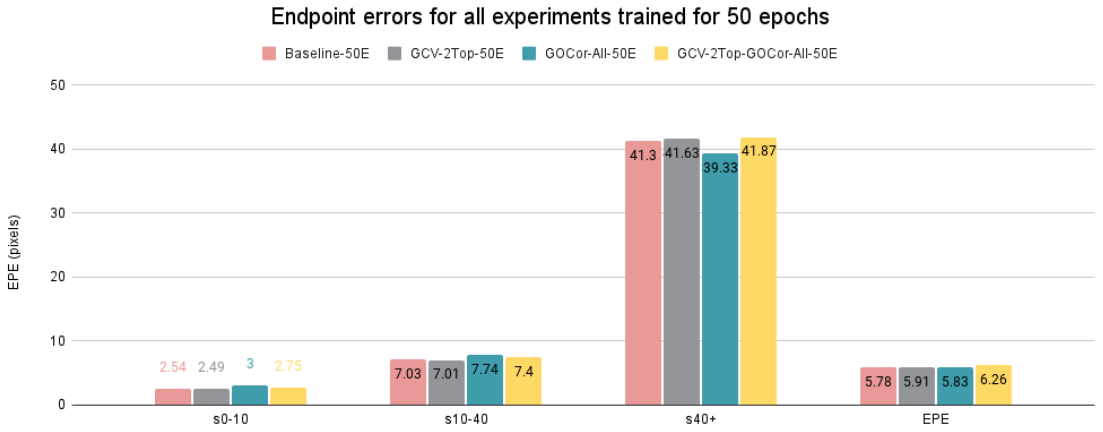

Figure 5.17: EPE for all the experiments

Interestingly, the local GOCor in all the pyramid levels also increase the endpoint error for pixels with velocities between 10 and 40, however, the average EPE does not show a substantial degradation.

Chapter 6

Conclusions

In this investigation, we successfully incorporated the enhancements proposed and assessed their influence on optical flow estimation, particularly for large displacements. Our conclusions are as follows:

1. The results for Baseline and GCV-2Top experiments show that there is a significant reduction of the EPE for large displacements with the increase of the number of training epochs.

2. The sole use of global cost volume in the two higher layers of the feature pyramids does not yield improvements in the endpoint errors.

3. The sole use of local gocor in all the pyramid layers leads to an improvement in the estimation of optical flow for large displacements of 1.97 pixels, without a significant degradation of the average EPE.

4. The concurrent utilization of GOCor and the global cost volume in the two higher layers of the feature pyramid results in a deterioration of optical flow estimation outcomes, refuting the hypothesis posited in this study.

5. A GPU with 12GB of memory is not enough to train the UFlow model augmented with global cost volume and GOCor, a GPU of 24GB is recommended

instead. In contrast, a GPU with 12GB of memory is adequate for training and evaluating the model that integrates only the global cost volume in the two higher layers of the feature pyramids.

6.0.1 Future work

We believe that an interesting avenue for a future study is the impact of variable window sizes to find the optimal search window for each pyramid level.

We expected that increasing the window size, would reduce the optical flow estimation error for large displacements, but the results indicate that no benefit is obtained by this change. Perhaps this result indicates that each pyramid level can benefit from a different window size, instead of a uniform larger window size. In higher levels of the pyramid, the resolution of the image is lower, and a larger window might increase the number of mismatched pixels. Therefore, a smaller window size in the higher layers, might provide better results. In lower levels, the window size increase might not lead to an increase in the pixel correspondence mismatches. Hence, we can explore the appropriate window size for each pyramid level, and the impact on the optical flow estimation.

Bibliography

[1] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari. The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches. 2018.

[2] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7577 LNCS(PART 6):611–625, 2012.

[3] E. R. Davies. *Computer Vision: Principles, Algorithms, Applications, Learning: Fifth Edition*. 2017.

[4] A. Dosovitskiy, P. Fischery, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:2758–2766, 2015.

[5] B. K. Horn and B. G. Schunck. Determining Optical Flow, 1981.

[6] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet_2.0_Evolution_CVPR_2017_paper. *Cvpr2017*, pages 1–9, 2017.

[7] J. Jeong, J. M. Lin, F. Porikli, and N. Kwak. Imposing Consistency for Optical Flow Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022-June:3171–3181, 2022.

[8] R. Jonschkowski, A. Stone, J. T. Barron, A. Gordon, K. Konolige, and A. Angelova. What Matters in Unsupervised Optical Flow. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12347 LNCS:557–572, 2020.

[9] J. Kim, Donghyung and Cho, Sanghyup and Jeong. A Motion Vector Recovery Algorithm for Temporal Error Concealment using Optical Flow in H.264 Video Coding. *IEEE International Conference on Multimedia and Expo*, pages 1713–1716, 2006.

[10] B. A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6):84–90, 2012.

[11] M. Leordeanu. *Advances in Computer Vision and Pattern Recognition Unsupervised Learning in Space and Time A Modern Approach for Computer Vision using Graph-based Techniques and Deep Neural Networks*. Springer, 2020.

[12] P. Liu, I. King, M. R. Lyu, and J. Xu. DDFlow: Learning optical flow with unlabeled data distillation. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 8770–8777, 2019.

[13] P. Liu, M. Lyu, I. King, and J. Xu. Selflow: Self-supervised learning of optical flow. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:4566–4575, 2019.

[14] K. Luo, C. Wang, S. Liu, H. Fan, J. Wang, and J. Sun. UPFlow: Upsampling Pyramid for Unsupervised Optical Flow Learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1045–1054, 2021.

[15] N. Malcolm and J. J. Gibson. The Perception of the Visual World. *The Philosophical Review*, 60(4):594, 1951.

[16] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:3061–3070, 2015.

[17] N. Onkarappa. *Optical flow in driver assistance systems*, volume 13. 2014.

[18] L. Porzi, M. Hofinger, I. Ruiz, J. Serrat, S. R. Bulo, and P. Kontschieder. Learning multi-object tracking and segmentation from automatic annotations. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6845–6854, 2020.

[19] R. Saborio. Reconstrucción rápida de objetos tridimensionales. *Tesis para el grado de Magister Scientiae en Ciencias de la Computación*, 2013.

[20] L. Sevilla-Lara, Y. Liao, F. Güney, V. Jampani, A. Geiger, and M. J. Black. On the Integration of Optical Flow and Action Recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11269 LNCS:281–297, 2019.

[21] X. Shi, Z. Huang, W. Bian, D. Li, M. Zhang, K. C. Cheung, S. See, H. Qin, J. Dai, and H. Li. VideoFlow: Exploiting Temporal Cues for Multi-frame Optical Flow Estimation. 2023.

[22] A. Stone, D. Maurer, A. Ayvaci, A. Angelova, and R. Jonschkowski. SMURF: Self-Teaching Multi-Frame Unsupervised RAFT with Full-Image Warping. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3886–3895, 2021.

[23] D. Sun. Secrets of Optical Flow Estimation and Their Principles Optical flow : motion of image pixels. *Cvpr*, pages 2432–2439, 2010.

[24] D. Sun, X. Yang, M. Y. Liu, and J. Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume D, pages 8934–8943, 2018.

[25] R. Szeliski. *Computer Vision (Texts in Computer Science)*, volume 42. 2010.

[26] S. Tafseer, H. Shah, and X. Xuezhi. Traditional and modern strategies for optical flow : an investigation. *SN Applied Sciences*, 3(3):1–14, 2021.

[27] Z. Teed and J. Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12347 LNCS:402–419, 2020.

[28] P. Truong, M. Danelljan, L. van Gool, and R. Timofte. GOCor: Bringing globally optimized correspondence volumes into your neural network. *arXiv*, (NeurIPS), 2020.

[29] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. *Proceedings of the IEEE International Conference on Computer Vision*, (Section 2):1385–1392, 2013.

[30] G. Wolberg. *Digital Image Warping*. Wiley, 1990.

[31] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao. GMFlow: Learning Optical Flow via Global Matching. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022-June:8111–8120, 2022.

[32] J. J. Yu, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9915 LNCS:3–10, 2016.

[33] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. M. Reid. Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018.

[34] F. Zhang, O. J. Woodford, V. Prisacariu, and P. H. Torr. Separable Flow: Learning Motion Cost Volumes for Optical Flow Estimation. *Proceedings of the IEEE International Conference on Computer Vision*, 10807:10787–10797, 2021.

[35] Y. Zhu, Z. Lan, S. Newsam, and A. G. Hauptmann. Guided Optical Flow Learning. 2017.