Space-time Stereo From Consecutive Stereo Pairs

Maksymilian Mozolewski

4th Year Project Report
Artificial Intelligence
School of Informatics
University of Edinburgh

2022

Abstract

There are many deep machine learning models that compute depth from binocular stereo images, and many models which compute Optical Flow from monocular video. With calibrated cameras, both can give dense 3D scene descriptions. While these approaches perform very well, they do not take advantage of temporal coherence of stereo video. We develop and explore 8 different architectures which take as input two consecutive pairs of stereo images and produce a dense disparity image. We design these while exploring two different modality combination paradigms, namely Early and Late Fusion of binocular video frames. Additionally, we perform rigorous evaluation and statistical analysis on the KITTI dataset (Mayer et al., 2016). We show one of our Late Fusion models is at least as and possibly more performant (albeit with weak statistical evidence) than LEAStero - a State of The Art stereo network. We also analyse error maps on select dataset samples and suggest improvements which could push performance even further, paving the way for future work.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Maksymilian Mozolewski*)

Acknowledgements

Thanks to Bob Fisher who has been an amazing supervisor, and Manaal my fiancée who has kept me sane.

Of course I couldn't have done this without my trusty 1080TI GPU, which has been running non-stop for many a day.

Table of Contents

Chapter 1

Introduction

Stereo Vision enables us to reason about our environment with 3D information, it has many benefits over other types of 3D sensors such as the lack of interference with other devices. Interference is a problem with active sensors like Lidar when the use of many systems in close proximity is required (Bertozzi et al., 2000). Binocular cameras can also sample the environment at much higher rates than Lidar, can generate complete (non-sparse) depth images, and are relatively cheap. The applications of Stereo Vision mostly lie within the area of self-driving cars, but also in assistive robotics (Molton et al., 1998) and in any field where 3D depth is required.

In this project we explore the combination of two State of The Art (SOTA) networks: one which generates Optical Flow given consecutive monocular video frames, and one which produces dense disparity data given a stereo pair of images (see section 2.2). Our goal is to produce temporally cohesive and more accurate disparity images. We approach the problem by designing multiple architectures (see section 3), and filtering them down to a final one (see section 4.4.1.1). We begin with networks which deviate from the SOTA networks the least and iterate on the design to progressively make greater changes.

We divide our designs into two:

- Early Fusion - Where we don't explicitly model Optical Flow, and video frames are fused into the model at an early stage.
- Late Fusion - Where Optical Flow is modelled explicitly and video frames are fused later into the model.

In addition to training all our models (see section 4.2), we also carry out statistical tests and rigorously evaluate our best performing models (see section 4.4). At the end we also analyse our final architecture in detail by looking at error maps, to identify possible improvements (see section 4.5).

We begin by describing in brief, the terms and definitions with which the reader should be familiar to follow the discussion.

1.1 Preliminaries

1.1.1 Stereo Vision

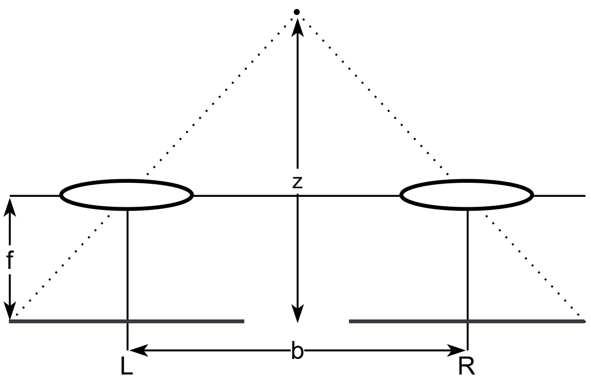


Figure 1.1: Stereo camera setup, the ellipses represent camera lenses, and L and R the rectangular image sensors seen from the top. The black dot represents an object seen by both cameras, and it is projected to the same vertical coordinate on both resulting images. Ff the dot was translated in the left/right directions in the diagram, its projection on both images would trace out the epipolar line.

Stereo Vision is the calculation of depth from pairs of images taken with binocular cameras - an example stereo setup is shown in figure 1.1. This method requires two images I_l and I_r as well as a correspondence $C : \mathbb{R}^2 \nrightarrow \mathbb{R}^2$ from each pixel (x, y) in the left image (traditionally) to a pixel in the right image.

The images are assumed to be *rectified*, meaning that any object captured by both cameras, is projected to the same vertical coordinate - that is both of the camera's *epipolar lines* are horizontal (The projections of rays corresponding to each pixel on one camera onto another camera's image plane). If a pixel (x, y) in the left image, matches the pixel $(x - d, y)$ in the right image, then d is defined as the *disparity* in the left image. It is more favourable to infer disparity rather than depth directly, since disparity is detached from camera intrinsics. Depth is directly linked to disparity with:

$$z = \frac{bf}{d} \tag{1.1}$$

Where d is disparity, f is the focal length of the cameras, b is the *baseline* between cameras and z is the depth.

Calculating the correspondence C is non-trivial due to variations in lighting, position and shape between the images as well as depth discontinuities/occlusions which entirely disconnect regions of the pictures from each other (hence C is not a total function).

1.1.1.1 Traditional Stereo Vision Pipeline

Traditional methods for Stereo Vision usually handle the problem in 4 (or less) steps (Scharstein et al., 2002):

1. Matching cost computation: The computation of dissimilarity between possible corresponding pixels based on their intensities
2. Cost aggregation: The aggregation of costs over chosen support regions
3. Disparity computation: A selection of disparities according to some strategy based on the computed costs
4. Disparity refinement: Filtering of the final disparity to smooth out the image or enforce additional constraints

As an example, a simple approach would be to minimize the squared difference of intensities between matches:

$$Cst(p^l, p^r) = \sum_{(l,r)\in N(p^l,p^r)} (I_l(l) - I_r(r))^2 \tag{1.2}$$

Where $N(a,b)$ are some (spatial) neighbourhoods around the pixels a and b respectively, over which the cost is *aggregated*, and $I_i(a)$ is the intensity of pixel a in the image i.

That is, we try to find for each pixel p^l in the left image a pixel p^r in the right image, with the same vertical coordinate, for which $Cst(p^l, p^r)$ is the smallest.

In this case 1 and 2 could be thought of as a single step, and such disparity selection is formally named *Winner-Takes-All* (WTA) selection:

$$D(p^l) = \arg\min_d \; Cst(p^l, [p^l_x - d \quad p^l_y]) \tag{1.3}$$

Methods vary in how much of the contextual information they make use of, falling into one of 3 categories:

- Global methods: If they perform some form of optimization over the whole image
- Local methods: If they optimize over relatively small local regions
- Semi-global methods: If they perform a combination of both

1.1.2 Space-time Stereo

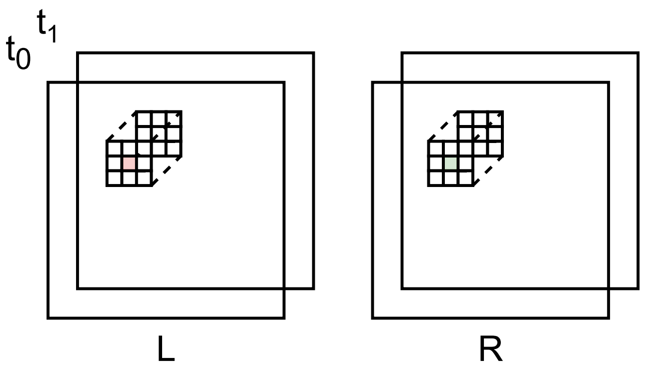


Figure 1.2: Space-time Stereo, an additional pair of images is added representing a second point in time. Matching is performed across both space and time dimensions.

Equation 1.2 can be improved for certain images, by expanding the neighbourhood N to contain *temporally* adjacent pixels (shown in figure 1.2).

Davis et al. (2003) defined Space-time Stereo as a broad framework, which encompasses all stereo methods. Under this framework, no attempt is made to derive 3D motion. Instead cost windows are sheared and translated, according to the predicted motions in the image and Optical Flow. In this project we use this term to define methods strictly making use of at least 2 frames of binocular video. Those which use solely spatial neighbourhoods, are referred to as Stereo Vision methods.

The temporal axis provides additional information which helps when lighting or even the scene itself is dynamic (flickering lights, or moving cameras) if the Space-time window is modelled non-statically (L. Zhang et al., 2003). It can also be helpful in ill-posed regions such as occlusions, and depth discontinuities.

1.1.3 Optical Flow

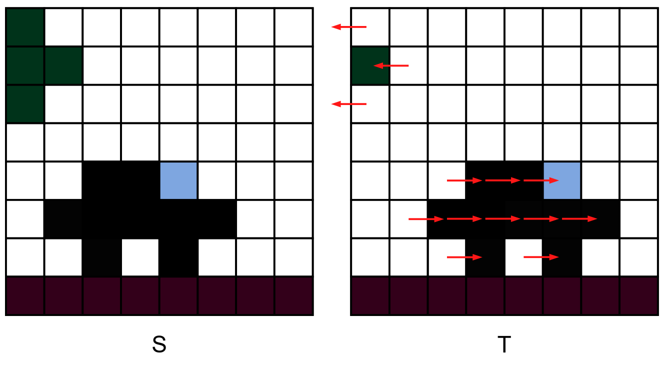


Figure 1.3: Optical Flow from the source image S to the target image T, overlayed over the target image

Optical Flow (Horn et al., 1981) is a method of quantifying the apparent motion in an image. The goal of Optical Flow methods is to attach a velocity vector δ for each pixel p^s in the source image, such that the following, *brightness constancy* assumption holds:

$$I_s(p^s) = I_t(p^s + \delta) \tag{1.4}$$

Occlusions cause similar problems to those found in Stereo Vision, however no epipolar constraints can be derived, and so each pixel's motion is only constrained by the dynamics of the scene and camera.

Optical Flow can be thought of as a projection of the 3D movement of objects in the scene onto the source image. Typically traditional methods formulate Optical Flow as a global optimization problem, with the objective of minimizing the violation of brightness constancy, and introduce additional constraints (like smoothness) to make the problem solvable.

1.1.4 Need For Deep Learning

Traditional algorithms, such as the ones shown above, fail in image regions which require prior knowledge and semantic understanding of the objects present.

The pitfalls of traditional methods are where deep learning is most powerful - the ability to learn from large quantities of data. Convolutional Neural Networks (CNN's) manage to extract features (Lecun et al., 2000), and match them in a way that traditional algorithms cannot. This is why most modern stereo research is based around the applications of CNN's, and why the top places in stereo benchmarks are occupied by deep learning approaches.

1.2 Contribution

This project explores the incorporation of Space-time methods into SOTA deep learning approaches. We hypothesise that combining methods from Optical Flow and Stereo Vision should allow us to produce temporally coherent disparity maps of higher quality, than those produced using either approach by itself.

We design and evaluate 8 different architectures, one of which achieves 0.012 % smaller error on 5-fold cross validation over the KITTI dataset than LEAStereo - the SOTA stereo network we build upon. While this result is statistically insignificant at a p-value of 0.094, we propose improvements which could push performance higher, and show that our model was disadvantaged during training due to hardware restrictions.

1.3 Previous Work

Deep Space-time Stereo has been largely unexplored - perhaps due to the way stereo datasets are structured.

Deep Scene Flow methods are the closest to our approach, however they aim to solve a naturally much more difficult problem. Instead of trying to model 3D motion between frames, Space-time Stereo methods assume a linear model of motion over matching windows; We therefore attempt to incorporate temporal information into a deep stereo network to help resolve ambiguities and produce smoother disparities over videos.

Below we discuss work related to Stereo Vision, Optical Flow and Space-time Stereo:

Stereo Vision The first use of deep learning in Stereo Vision was done by Zbontar et al. (2015), who used a Convolutional Neural Network (CNN) to perform cost matching between 9x9 image patches. This proved successful, yet also computationally inefficient. This approach failed to utilize global learn-able patterns in the data. The first deep stereo network to do that was *DispNet* (Mayer et al., 2016) - this architecture however directly regressed depth information from the image pair, and the alternative model *DispNetCorr* used a correlation layer which collapsed the feature dimension, and did not capture context well. This data-driven approach required a large amount of resources to train. Kendall et al. (2017) overcame this by concatenating unary features from both images over the disparity dimension, in an effort to replicate traditional cost volume based approaches. Their novel *GC-Net* performed better, and converged quicker. Recently Cheng et al. (2020) applied Hierarchical Neural Architecture Search (NAS) to optimise GC-Net, creating a SOTA network - *LEAStereo* with only 1.81M parameters. Approaches incorporating similar cost volumes to *GC-Net* became popular.

A lot of other recent work also improved performance, however often at the cost of inference time or parameter count. Chang et al. (2018) incorporated Spatial Pyramid Pooling (SPP) and intermediate supervision via a stacked hourglass matching network which reduced error in ill-posed regions (texture-less areas, occlusions etc.). Then F. Zhang et al. (2019) presented a semi-global cost aggregation layer, which was able to replace 3D convolutions at a smaller cost and higher accuracy. Guo et al. (2019) proposed a group-wise correlation cost volume which combined the benefits of a feature correlation layer, and a concatenation layer, decreasing inference time and parameter counts. At time of writing Mao et al. (2021) proposed to sample candidate disparities sparsely based on uncertainty distributions and learned per-pixel disparity ranges, which is more efficient.

All of the work presented above seeks to improve performance using spatial information present in the left and right images. In our project, we build directly on *GC-Net* and *LEAStereo* to investigate if explicitly incorporating temporal coherency found between consecutive stereo frames, would yield gains in performance, even in already well performing networks. *LEAStereo* is a great baseline and foundational model for this project, due to its small parameter count and clear, interpretable structure - because this network is built for Neural Architecture Search (NAS), it opens up the possibility for searching over the network structure to further improve results.

Optical Flow Dosovitskiy et al. (2015) introduced the first successful end-to-end Optical Flow network - *FlowNet*, however it couldn't outperform traditional methods. The network was trained on a dataset which was largely unrealistic. Ilg et al. (2017) presented an improved version of *FlowNet*, they combined networks which operated over smaller displacements, and fused their outputs, while training on an improved dataset along with the original one.

Soon after Sun et al. (2018) designed a much smaller *PWC-Net* which introduced a warping and cost volume layer, on top of feature pyramids. This reduced inference time and performance drastically.

Zachary et al. (2020) pushed performance even further, by first building 4D correlation volumes, and iteratively updating Optical Flow fields via correlation look ups. At time of writing at least 13 networks within the top-30 in the KITTI 2015 benchmark are derivatives of this network.

Space-time Stereo Space-time Stereo was initially introduced as a framework by Davis et al. (2003), and refined concurrently by L. Zhang et al. (2003) to handle dynamic scenes. For static scenes, matching across multiple frames, allowed for the shrinking of spatial windows without sacrificing performance, while in general it helped to handle changes of texture, lighting and motion. However, the original Space-time method, was a local technique and hence produced disparity maps which were not very smooth. L. Zhang et al. (2004) presented a globally consistent Space-time Stereo formulation, which incorporated smoothness constraints for neighbouring pixels.

Bleyer et al. (2011) projected support windows onto disparity planes in order to avoid fronto-parallel bias, while incorporating temporal coherency. Noticing that matching in the temporal direction is more sensitive to motion Nover et al. (2018) improved on Bleyer et al. (2011). They proposed the Space-time descriptor *breve* which alternates between matching pixels: in the same frame, and across many frames.

Tsekourakis et al. (2019) empirically showed that enforcing temporal coherency yields better results in traditional methods. Tian et al. (2019) attempted to incorporate Optical Flow and disparity maps from previous frames, to enforce the concepts of 'reliable temporal neighbourhoods', pruning pixels which are too dissimilar from their matching pixels in the previous frame. This method ranked 10th on the KITTI 2015 stereo benchmark at the same time that *GC-Net* ranked 2nd. Most noticeably, all the other methods within the top 10 were deep learning approaches; However this method was about a 130 times slower.

Our proposed approach aims to combine the fast speed, and learning ability of neural networks, with traditional Space-time Stereo approaches, in order to incorporate a form of temporal coherency into disparity maps as well as to hopefully improve disparity quality.

Chapter 2

Background

Deep Space-time Stereo has potential to improve upon existing deep Stereo methods in two main ways:

- By creating temporally coherent disparity maps
- By reducing error at object boundaries and discontinuities

Before we present our networks attempting to incorporate Space-time methods, we must first introduce the benchmark and datasets used as well as the models on which we build on.

2.1 Datasets

Training Stereo and Optical Flow networks requires vast amounts of labelled training data. Additionally, for our Space-time networks all training samples need to contain at least 2 consecutive frames of stereo video along with ground truth disparity data.

Due to limitations on hardware and time, in this project we make use of transfer learning to shorten training times. We base our approach on that of Cheng et al. (2020), i.e. initially training on a much larger synthetic dataset, and then fine tuning on a target dataset upon which bench-marking is performed. However wherever possible we initialize our models with pre-trained weights (found in previous work) thus removing the need for the longer initial training step.

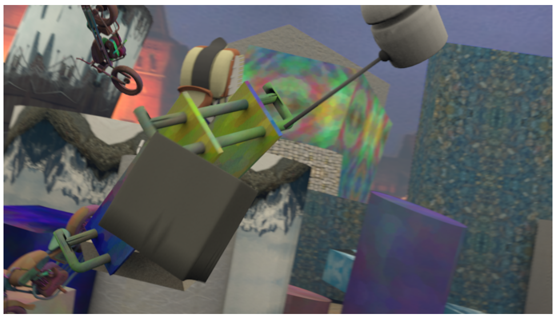
Below we detail the specific datasets involved in the training or evaluation of our models.

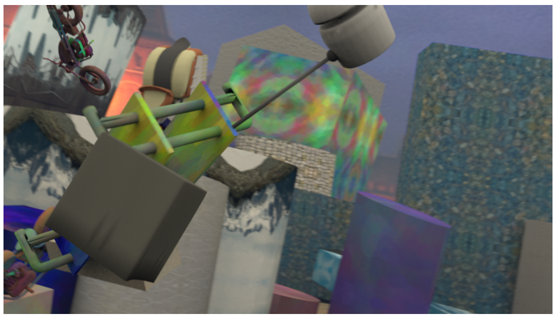
2.1.1 FlyingThings3D

FlyingThings3D also known as Sceneflow (Mayer et al., 2016), contains scenes of randomized textured objects flying in randomized directions. It contains more than 20 thousand labelled samples in 960 x 540 resolution, each with left, right and disparity images.

While Sceneflow, is synthetic and unrealistic, it is very large. Large realistic datasets are expensive to acquire, but given a dataset like Sceneflow and another small but realistic dataset, we can use transfer learning to perform well on the smaller dataset.

Samples from Sceneflow can be seen in figure 2.1.


(a) Left image


(b) Right Image

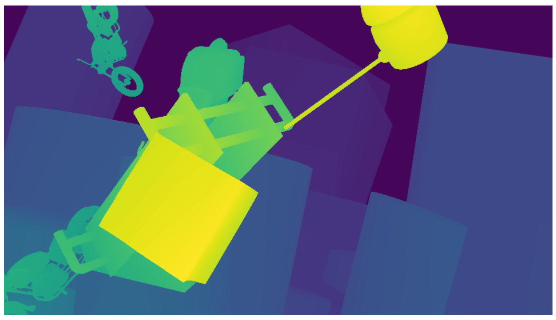
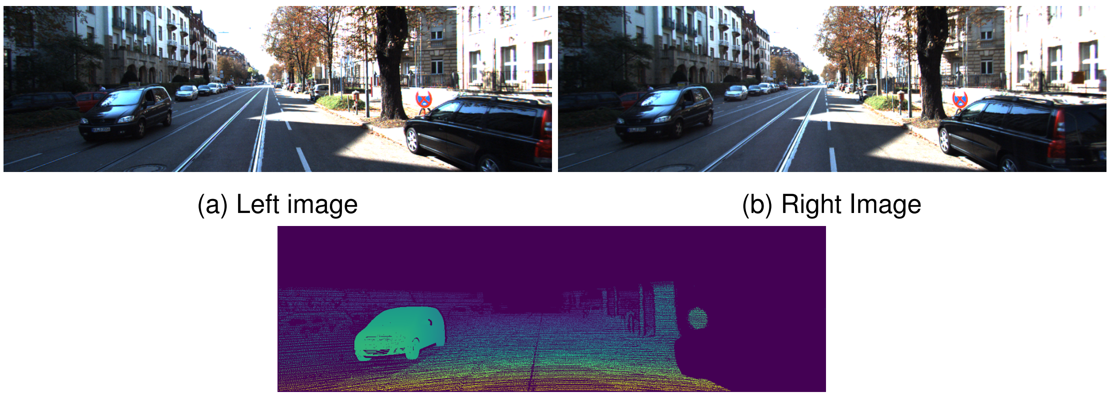

(c) Disparity in the left image

Figure 2.1: Sample frames from the FlyingThings3D dataset

2.1.2 KITTI 2015

The benchmarking dataset we use is KITTI-2015 (Mayer et al., 2016). It contains 200 sequences of 1242px by 375px images captured by a stereo camera mounted on a vehicle equipped with a Lidar. This setup allows for generation of ground truth disparities between left and right images, however due to the sparsity of the data measured by the Lidar, the ground truth itself is 50% sparse. Samples can be seen in figure 2.2.

KITTI images are relatively small in their resolution, and a wide body of literature is evaluated on it, which makes it a perfect fit for this project.



(a) Left image (b) Right Image

(c) Disparity in the left image

Figure 2.2: Sample frames from the Kitti 2015 dataset. Note the disparity image is 50% sparse due to the data collection method.

We note that all ground truth labels (i.e. disparities) are expressed in the frame of reference of the left image at the first time step of each sample.

2.2 Models

As part of the project, we selected a network capable of producing disparity maps, and a network capable of generating Optical Flow. It is essential that the networks are relatively small, so they can be trained on a single GTX 1080TI GPU, but also generally simple, so that modifications to their architecture are easy to interpret and execute.

We selected LEAStereo and RAFT for these purposes and we detail the reasons and their architectures below.

2.2.1 LEAStereo

LEAStereo stands for *learning effective architecture stereo*, it was designed to be as small as possible in order to make hierarchical neural architecture search over it feasible. This network is currently ranked third on the global KITTI benchmark in spite of the small number of parameters. The architecture (shown in figure 2.3) builds upon *GC-Net*, borrowing the idea of packing features into a 4D feature volume which is then reduced into a 3D cost volume, and minimized using a Softmin operation.

The network first extracts features via a smaller *feature network*, with shared parameters for both images, then concatenates the features along a new disparity axis, at the same time shifting each map by their disparity width wise. The larger *matching network* takes the feature volume and computes a 3D cost volume with 3D convolutions (across the spatial as well as the disparity axes), from which the final disparity map is inferred via a softmin operation (i.e. smallest cost is picked).

The feature volume is constructed using a third of the maximum disparity levels, and then up-scaled via interpolation before softmin, to reduce model size.

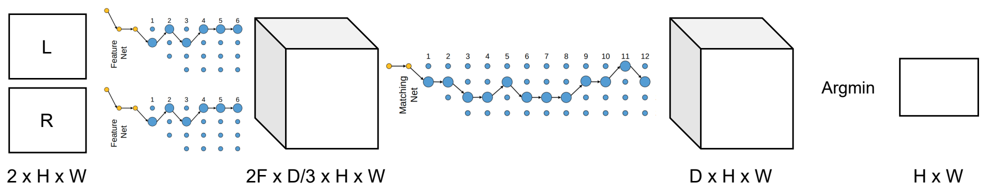


Figure 2.3: LEAStereo model architecture, the left and right images are fed into the network, and a single disparity image is produced. D is the maximum disparity used, and F the number of features extracted by the feature network. Parts of this figure were adapted from Cheng et al. (2020)

Each sub-network, is composed of *Feature* and *Matching* Cells, which receive as input the features from 2 previous layers, and combine them with skip connections and convolutions. The order of, and operations themselves were searched as part of the NAS in the original paper for each cell type and can be seen in figure 2.4.

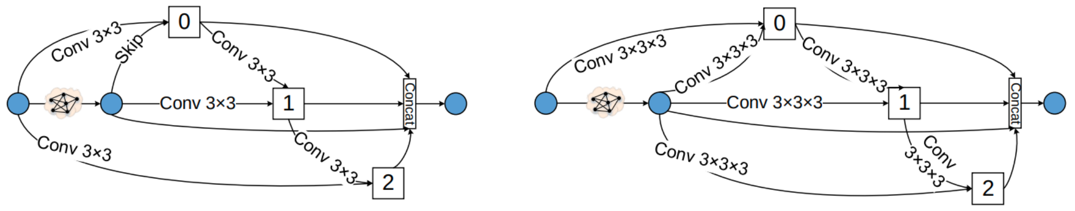


Figure 2.4: On the left, Feature Cell architecture, on the right Matching Cell architecture, first 2 nodes (in blue) represent the 2 previous sets of features. Where a node receives more than one input, they're summed. Inputs are interpolated to match the scale at which the cell operates, and feature counts changed appropriately with convolutions. Images from Cheng et al. (2020)

Each cell can operate on a different scale of the original resolution (1/3,1/6,1/12,1/24), and the individual scales at each layer were part of the search also. The number of features extracted is doubled each time the resolution is halved.

During training, random cropping and per-sample image normalization are the only data augmentation methods used.

2.2.1.1 Reproduction

The original model, was designed primarily with NAS in mind, and is hardly adaptable or extensible, in order to build on top of LEAStereo, we re-implement the model. Before proceeding with any experiments our version of code had to be verified against the original model, and so models were trained on both versions of the code on the KITTI 2015 dataset using pre-trained weights from the Sceneflow dataset, with three different seeds. Performance on a validation set (20 samples) was passed through a Wilcoxon test, and models were confirmed to be statistically similar. The validation performance can be seen in figure 2.5

In addition, the final weights from the original authors were evaluated on our test rig, to verify that our metrics and network produced correct results, and that both networks produced the exact same output given the same inputs. These results can be seen in figure 2.6

The model was developed alongside a robust and general framework designed to be able to run many models with different training and test time needs from a single codebase. The test rig allowed different transformations to be applied to the test data by different models, as well as swapping or randomizing inputs completely (for feature importance tests)

Method	Epochs	Batch size	Learning Rate	Crop size	Error (%)
LEAStereo (ours)	400	4	1e-3	336x168	5.662 ± 0.350
LEAStereo	400	4	1e-3	336x168	5.872 ± 0.145

Figure 2.5: Experiments run to verify baseline implementation, Error is an average taken over 3 separate training runs over different seeds. Networks were trained from the same pre-trained Sceneflow weights, and trained for a further 400 epochs.

Method	Error (%)
LEAStereo (ours)	1.880
LEAStereo	1.880

Figure 2.6: Experiments run to verify test rig correctness, the networks were evaluated after being initialized to the same pre-trained KITTI weights, and produced identical results.

Next we explore our selected Optical Flow network.

2.2.2 RAFT

RAFT (Zachary et al., 2020) stands for Recurrent All-Pairs Transforms, and it generates Optical Flow using an iterative approach. This model currently ranks 38th on the KITTI Optical Flow benchmark, and while not in the top-10, it achieves only 2% larger error compared to the top ranking network, while being 10 times faster. At least 13 networks within the top-30 in this benchmark, are derivatives of this network.

The network consists of a feature encoder, a context encoder, and the GRU recurrent update operator. Left and right features are then correlated pixel against pixel using an inner product, and these correlations stored as 4D Volumes. These are then pooled to multiple scales and fed into the update operator via look-ups (with different scales concatenated), together with the output of a context encoder.

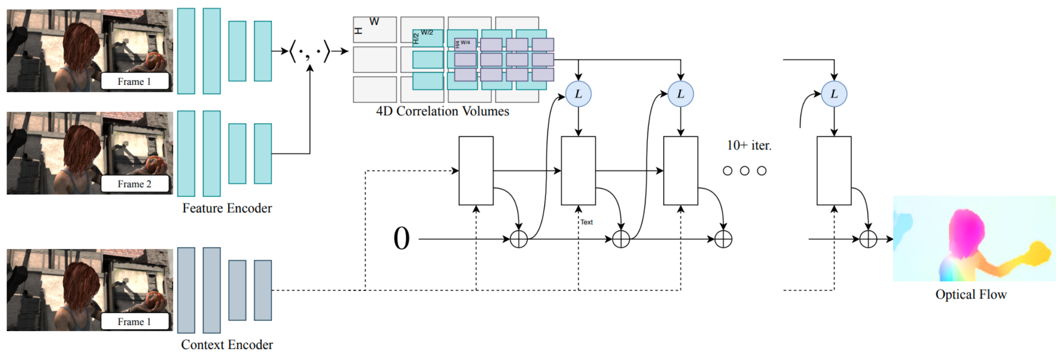


Figure 2.7: RAFT network architecture, two consecutive images from a video are fed into the network, which then produces Optical Flow from the first to the second frame. Image from Zachary et al. (2020)

The update operator can be either initialized with zeros or a previous Optical Flow image, which is a very useful property.

2.2.2.1 Reproduction

RAFT was not trained in this project, since it was used as an isolated pre-trained sub-component. Therefore no adaptations were necessary to the original network, and the network produced expected results on the KITTI dataset.

Chapter 3

Architectures

We divide our approaches into two major avenues: Early and Late Fusion of time frame data. We either try and inject mostly unstructured time features into the early parts of the network, or process them separately producing structured output, which we then combine into some later part of the main network. Both approaches have advantages and drawbacks.

With Early Fusion, a lot more common processing can be performed by the network, i.e. both Optical Flow and Stereo Vision perform matching between two images in order to find correspondences, with the key difference that in Stereo Vision, those correspondences are constrained to occur in a single epipolar line. Early Fusion is more prone to make use of such inter-task commonality and hence reduce repeated processing of features. However, Early Fusion results in features which might be sub-par for either of the tasks and make optimization harder.

Late Fusion is mainly advantageous in that it promotes abstraction and compartmentalization of the network - features are processed separately until they can be interpreted and used. This method can introduce a lot of duplicate processing, but it gives rise to networks which can have certain parts re-placed and trained separately, reducing memory use when training the main network.

As part of this project we attempt to build models with both of these methodologies, in search of the best solution. All models presented here, have LEAStereo as the backbone.

We designed and trained four main Early Fusion models and two Late Fusion models iteratively. We detail their architectures in following sections.

Each model is named after the main modification introduced to the original architecture. All model names begin with: ”STS” which stands for - Space-time Stereo, followed by either: ”EarlyFusion” or: ”LateFusion” and then the network’s suffix.

3.1 Early Fusion

The main parts of the network being modified for our Early Fusion models, are:

- The feature volume
- Feature processing
- The outputs

All the networks presented are attempts at establishing an effective Space-time feature generation and processing pipeline. The main difference to the Late Fusion models presented later, is the lack of a designated Optical Flow network, since we believe that this approach should allow for more efficient use of common features.

However this approach also requires the network to learn a warping mechanism G:

$$G(I_{t1}) = I_{t0} \tag{3.1}$$

Which transforms an image seen in the second time step, to the corresponding image in the first (e.g. $left_{t1} \rightarrow left_{t0}$). Note we build such a mechanism directly into our Late Fusion models in section 3.2

3.1.1 STSEarlyFusionConcat (STS-EFC)

LEAStereo accepts two images as input, Space-time networks will all require at least four inputs, i.e. two pairs of images, at consecutive time steps. A straight forward extension of the 4D feature Volume, is then to simply concatenate all image features extracted by the feature network along the feature dimension, and pack them into the feature volume exactly the same way it is done in LEAStereo. This feature volume can then be put through the standard matching network, with the number of input filters doubled. This can be seen in figure 3.1.

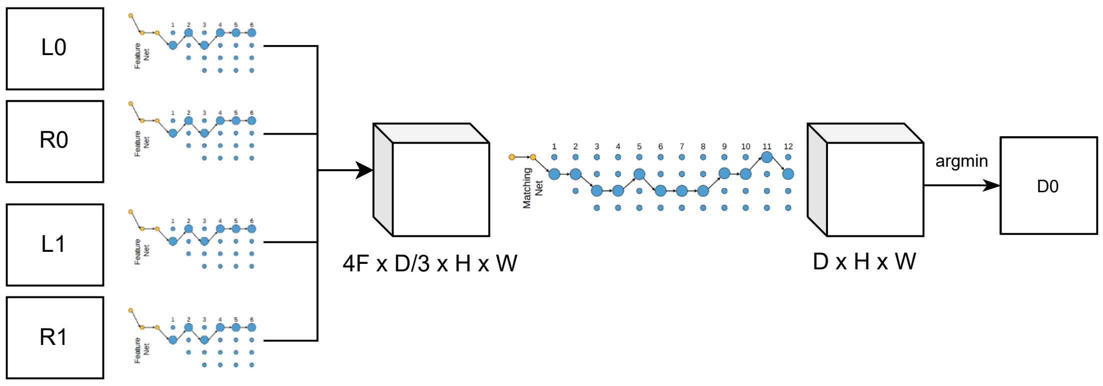


Figure 3.1: STSEarlyFusionConcat architecture. The architecture only differs from LEAStereo in the The number of inputs and features features (F) in its feature volume, which are both doubled. L1 and R1 are additional inputs, and correspond to the left and right images from the second time step respectively.

An issue with this design however is that, since the matching network operates with 3D convolutions, the initial convolution does not perform any cross-correlation or derive any inter-image features, in effect failing to capture any sort of flow between the time steps. In addition the output of the network is not related to the second time step whatsoever, since only disparity at time 0 is calculated. This means no loss term exists to force the network to use the features of images from the second time step.

3.1.2 STSEarlyFusionConcat2 (STS-EFC2)

A natural upgrade over the STS-EFC model is the addition of a second disparity output, this alleviates the issue of ignoring features by forcing the network to make use of the secondary time step. The loss criterion is modified to be the mean loss from both outputs weighted by their importance as follows:

$$Loss = \alpha \mathcal{L}(D0) + (1-\alpha)\mathcal{L}(D1) \tag{3.2}$$

Where α is a mixing factor set to 0.75 to prioritize the accuracy on the first disparity, and $\mathcal{L}$ is the specific loss function being used. For our training we used smooth L1 loss (Girshick, 2015).

This network can be seen in figure 3.2.

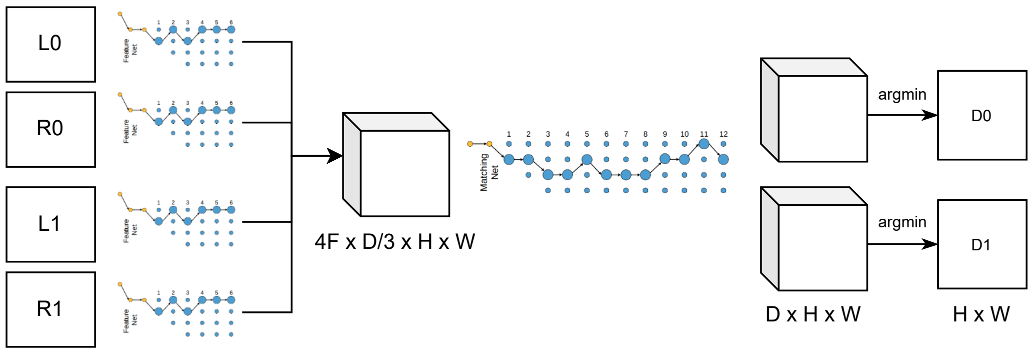


Figure 3.2: STSEarlyFusionConcat2 architecture. A second disparity D1 is generated corresponding to the second time step.

This arrangement should at least force the network to make use of both sets of spatial features, individually (i.e. without correlating across time steps).

3.1.3 STSEarlyFusion4D (STS-EF4D)

While STS-EFC2 fixes one of the big problems of STS-EFC, it still doesn't capture temporal features very well, A plausible solution to this problem is to pack all the features into two feature volumes as normal, one per time step, and then reduce this to a single feature volume with a 4D matching network aiming to generate cross-time features first and then match using those as well. This concept can be seen in figure 3.3

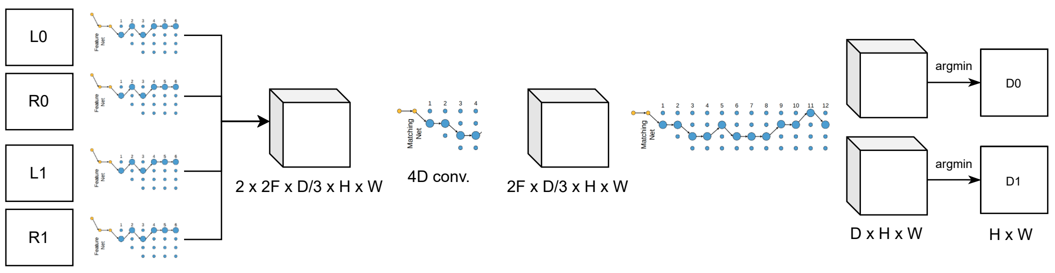


Figure 3.3: STSEarlyFusion4D architecture. A 4D convolution matching network is inserted before the 4D feature volume, and a 5D feature volume is added.

While in theory this is a natural and straightforward extension, in practice however, 4D convolutions consume high amounts of memory, hence training them is intractable in our setup and this architecture is left untested.

3.1.4 STSEarlyFusionTimeMatch (STS-EFTM)

Another way to extract temporal features is to inject a temporal matching network between the feature volume and across the time steps as seen in figure 3.4. Instead of using 4D convolutions, 3D convolutions are used here. In theory such convolutions should be able to match patterns along the feature axis, analogously to the main matching network but in this case the matching axis is temporal and not spatial. Each temporal matching sub-network, produces additional temporal features which are packed into the feature volume in the same way this is done for STS-EFC (the output shape is 1 x F x H x W, i.e. the temporal axis is reduced to a single dimension via convolutions).

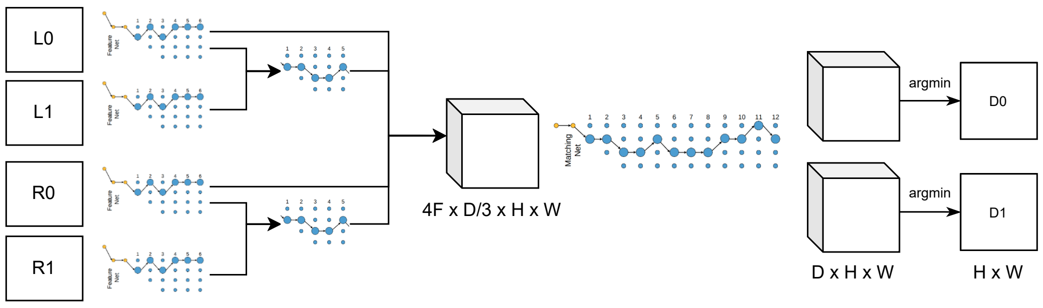


Figure 3.4: STSEarlyFusionTimeMatch architecture. Additional 3D convolution matching networks are inserted in between left and right images across time steps.

3.1.5 STSEarlyFusion2Big (STS-EFC2B)

One issue with simply doubling the inputs and outputs to the original network is the fact that now the network now is effectively bottle-necked by its matching network. To compensate for this we can increase its expressive power by doubling the amount of parameters. In this case we stack the network depth-wise as seen in figure 3.5.

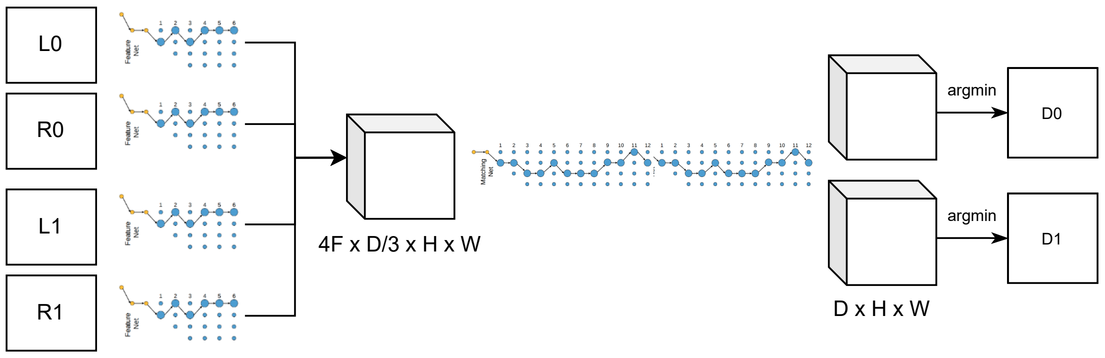


Figure 3.5: STSEarlyFusion2Big architecture. The 3D matching network is stacked depth-wise.

As explained previously, the models presented so far, must learn a warping mechanism G. This might be difficult and hence we now move on to Late Fusion architectures where such a mechanism is built directly into the network.

3.2 Late Fusion

Our Late Fusion models explicitly incorporate the RAFT Optical Flow network, and then warp the second disparity image to unify their frames of reference. However the warp G, requires an inverse mapping to that generated by forward Optical Flow, which must either be generated or calculated. No straight forward inverse operation exists for Optical Flow, due to its surjectivity, although one might train a RAFT-inverse network (see section 3.2.3). This surjectivity also implies that an inverse must be sparse and hence post-processing must be performed in order to produce smooth and complete warped images.

In all of our late Fusion models, we first generate two disparity maps, one for each time step and then attempt to unify and post-process them together to make use of the inherent temporal coherency between the time steps. In our networks we use an auto-encoder to refine the disparity maps and enable the network to share information and improve the disparity maps based on adjacent temporal information. While simple this mechanism should work as a proof of concept.

3.2.1 STSLateFusionGTFlow (STS-LFGTF)

To show that an improvement over the baseline is feasible, we designed a ground-truth-fed network. The basic Late Fusion model, employs an auto-encoder which takes as input two disparity images, one for each time step, both in the frame of reference of the first time frame (D0). These are then compressed and decompressed to force the model to learn some sort of mapping between the time steps, but an identity connection is made between the input and output of the auto-encoder, to avoid information loss. The final outputs of the network are two disparity maps as previously.

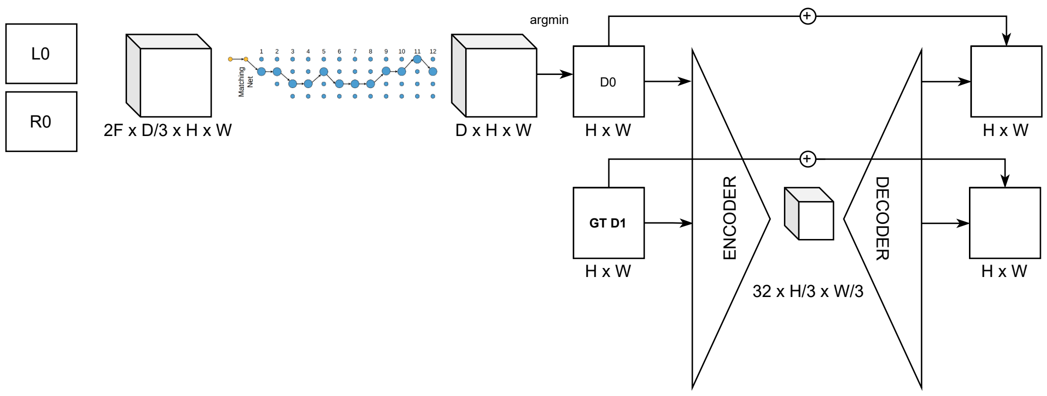


Figure 3.6: STSLateFusionGTFlow architecture. An auto-encoder and identity connections are used to combine ground truth disparity data for the second time step with the generated disparity for the first time step.

Should this network produce higher (D0) accuracy than the baseline, it would suggest that the model is able to extract this ground truth data from the second time step, and correlate it with the first through the auto-encoder.

3.2.2 STSLateFusion2 (STS-LF2)

To avoid using ground truth data, our model must produce the second disparity image, and then transform it such that it is within the same frame of reference as the first disparity image. We incorporate the RAFT Optical Flow network, to first generate the surjective forward mapping F:

$$F(I_{t0}) = I_{t1} \tag{3.3}$$

We then inverse this mapping, producing G the sparse backward Optical Flow, with which we warp the second disparity image into the first frame of reference:

$$G(I_{t1}) = I_{t0} \tag{3.4}$$

using the ”pseudo” inverse:

$$G(x,y) = -\lfloor F(x,y) + (x,y) \rfloor \tag{3.5}$$

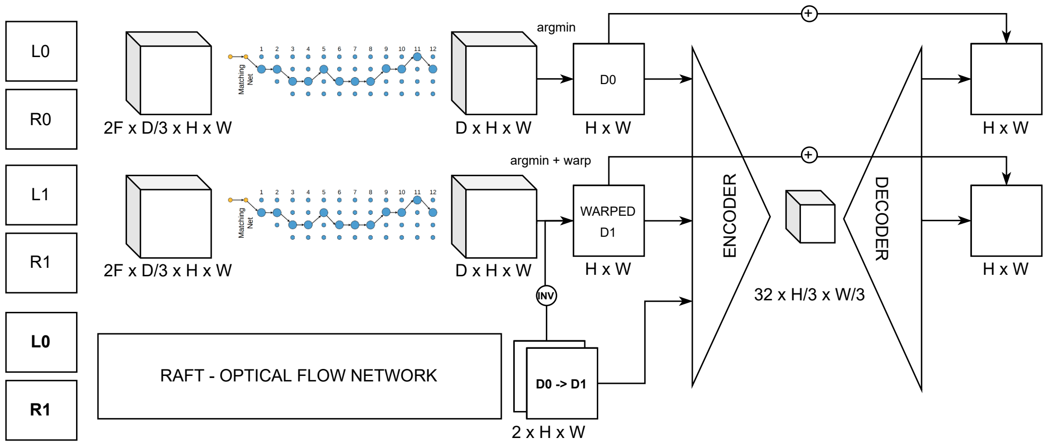


Figure 3.7: STSLateFusion2 architecture. The ground truth data is replaced with a second pass of the LEAStereo network, the disparity is then warped by the inverted Optical Flow from L0 to R1 and combined as previously through the auto-encoder.

Again, since $F(x,y)$ is surjective, not all pixels will receive backward flow, the gaps are filled in using a single pass of a median 3x3 filter. We note that the inverse operation is not differentiable, but since we use a pre-trained Optical Flow network, we do not need to propagate gradients through it. A diagram of this network can be seen in figure 3.7.

3.2.3 STSLateFusion2Inv (STS-LF2I)

An alternative to STS-LF2, is to infer the backward Optical Flow G directly via the Optical Flow network. This avoids the need for a inversion operation and allows for training of the entire network together. That is assuming the median filter is not used, since the backward warp is still required and will produce a sparse disparity map. This architecture can be seen in figure 3.8

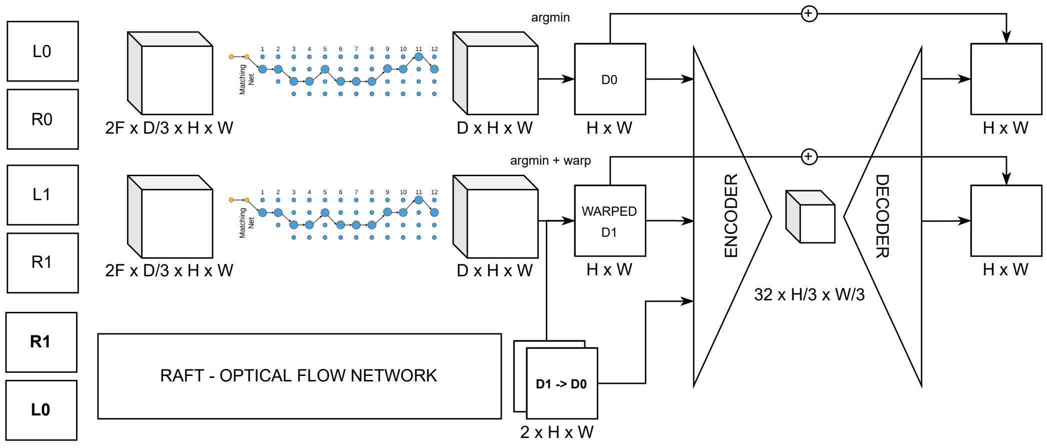


Figure 3.8: STSLateFusion2Inv architecture. Almost identical to STSLateFusion2 however, the order of RAFT inputs is inverted, meaning the inverted flow is generated directly, and then used to warp the generated disparity.

3.3 Summary

In this section we introduced 8 custom deep Space-time Stereo models, designed to make use of additional temporal information. All of them are plausibly improvements over the LEAStereo network. In chapter 4 we compare the performance of these models and identify those which have potential to outperform the baseline, and then verify if that's the case in thorough cross-validation.

Chapter 4

Results

Following the design of various architectures, we trained, evaluated and contrasted their performance to identify successful augmentations. We started with individual runs in order to select the most successful models, we then moved on to more thorough cross-validation, and trained some networks on multiple initial seeds.

4.1 Experimental Setup

We first describe our methodology:

4.1.1 Weight Initialization

All of our models weights were initialised to the weights provided by Cheng et al. (2020) (apart from RAFT), specifically the weights saved after LEAStereo was trained on the sceneflow dataset, and before it was fine-tuned on KITTI. This allows for an unbiased evaluation, but also reduces overall training time. One drawback of this approach is that the further our architecture was from LEAStereo, the more layers' weights had to be ignored during initialization, hence giving LEAStereo a slight advantage. We keep this in mind throughout the evaluation. RAFT is initialized directly to the weights provided in Zachary et al. (2020), specifically from the model fine-tuned on the KITTI dataset.

4.1.2 Early Stopping

We keep track of validation accuracy every epoch and at the end of the experiment, evaluate the model with weights which which produced the highest validation accuracy. An example training plot for LEAStereo can be seen in figure 4.1.

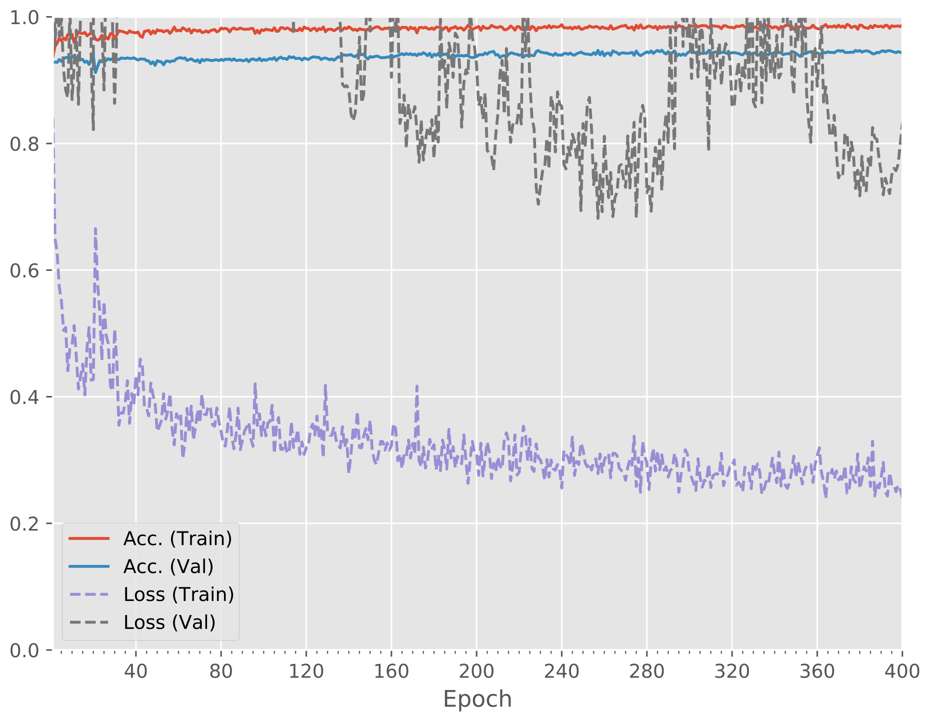


Figure 4.1: Plot of loss and accuracy (validation and training) from one of the LEAStereo training runs

We note that the the validation accuracy quickly rises in the first few epochs, then remains at a mostly stable level. Further improvements appear in later parts of the training, with the epoch distance between them increasing over time. We train all networks until the point where this distance is likely to be above a hundred epochs, and at that point we consider the model to have converged.

4.1.3 Hyper-parameters

We use similar hyper-parameters to Cheng et al. (2020):

- We use a learning rate of $1e-3$ and reduce it by half after 30, 50, and 300 epochs.
- Gradient descent is performed with the Adam optimizer with β_1 and β_2 set to 0.9 and 0.999 respectively.
- Images are normalized to unit mean and standard deviation per sample per channel.
- We randomly crop the images to 336 x 168 during training, and pad them to the 1248 x 384 at test time with the nearest pixel at the border. Cheng et al. (ibid.) use a higher resolution, however we were restricted by hardware.
- The maximum disparity (D) is set to 192.

- Batch size is kept as close to 4 as possible, but where a model is too big to fit in memory, smaller batch size is used.

4.1.4 Criteria

We use smooth L1 loss (Girshick, 2015) when training our models, unless otherwise specified. Output error is calculated by counting the number of disparities which are more than 3 pixels away from the ground truth similarly to Mayer et al. (2016).

4.1.5 Hardware Restrictions

Due to limited resources we are unable to train all our models on even ground. Some models are trained with smaller batch size, to fully fit into available GPU memory. We also use lower resolution than in the original work. Finally we do not fully pre-train all of our networks, meaning the networks which differ from the baseline the most, start off training at highest disadvantage. All of this is taken into account in our analysis of results.

4.2 Initial Experiments

We first trained each of the algorithms individually on a subset of 180 samples from the KITTI dataset then evaluated them on the other 20 samples (without a test set, in order to keep the training set as large as possible). These results can be found in table 4.1

Model	Epochs	Batch Size	Error
STS-LFGTF	425	2	5.111
STS-EFC2B	600	2	5.141
Baseline (LEAStereo)	425	3	5.658
STS-EFTM	425	3	5.660
STS-EFC	425	3	5.714
STS-LF2I	425	2	5.738
STS-LF2	425	2	6.111
STS-EFC2	480	3	6.135

Table 4.1: Our algorithms ordered by **validation error (%)**

An important result is the high performance of STS-LFGTF, this model has no access to ground truth for the first disparity frame (D_0) and therefore must have learned some correlation between D_0 and the second disparity frame (D_1) through its auto-encoder.

Overall two models improve upon the baseline: STS-LFGTF and STS-EFC2B. This result on its own does not show we have constructed a better model (because this was based on a single run), but acts as a proof of concept and motivates the rest of the project.

We also note that differences in weight initialization have played a factor as evident from the difference in performance between STS-EFC and STS-EFC2, as the latter should outperform the former.

4.3 Establishing Random Variance

Following individual experiments, we trained some of the models using different seeds, in order to establish the expected variance across individual runs due to randomness. This information is useful in analysis since it lets us predict if our augmentations had a real effect, or if the improvements are due to luck. We present the results in table 4.2

Model	Epochs	Batch Size	Mean Error
LEAStereo	400	4	5.662 ± 0.201
STS-EFC2B	600	2	5.402 ± 0.116

Table 4.2: Average **validation error (%)** and standard deviation of select models over 3 different starting seeds with the same test splits

With these results we can confidently say that due to randomness we should expect error to be at the very least within 0.2 % of the average value in the individual runs.

4.4 Model Comparison

We next move on to cross validation of the best performing models identified in the previous sections. We do this to isolate the effects of the differences in architectures, and to average out the effects of the noise of each particular dataset split. We use 5 non-overlapping test and validation splits, of size 40 and 20 respectively. Note that we increase the size of the test sets, since we found that a few samples are extreme outliers, hence why mean errors in table 4.3 are smaller than in the previous series of experiments.

Model	Mean Error
STS-LF2I	$\mathbf{2.656} \pm 0.292$
LEAStereo	2.668 ± 0.287
STS-LF2	2.731 ± 0.369
STS-EF2CB	3.075 ± 0.640

Table 4.3: 5-fold Cross Validation **test error (%)** and standard deviation of the best performing Early and Late Fusion models

We then performed paired (split-wise) Wilcoxon two-tailed statistical tests to confirm these speculations. We execute two granularity's of the test, by using either:

- Sample error - of each individual dataset sample
- Mean test set error - i.e. the error over each of the dataset test splits

We discuss these results below.

4.4.1 Late Fusion vs. Baseline

We notice that that STS-LF2I achieved a slightly lower mean error, however this improvement is statistically insignificant due to weak evidence ($p < 0.1$ as seen in table 4.4). Whereas STS-LF2 performed statistically worse. Tests at test split granularity however show that both models were indistinguishable from the baseline, which is to be expected due to low sample size.

These results can be seen in table 4.4, and histograms corresponding to each test can be found in figures 4.2, 4.3, 4.4 and 4.5

Test	Type	p-value
STS-LF2 vs. LEAStereo	Sample	0.002
STS-LF2 vs. LEAStereo	Mean	0.438
STS-LF2I vs. LEAStereo	Sample	**0.094**
STS-LF2I vs. LEAStereo	Mean	1

Table 4.4: Statistical significance test results

4.4.1.1 STS-LF2I

While small, the improvement certainly shows promise, and in light of the heavy disadvantages in training for STS-LF2I, potential to outperform LEAStereo on even ground. The training of this model was impeded due to:

- Smaller batch size of 2 due to memory limitations
- Sub-optimal starting weights, due to modifications to architecture and lack of resources to carry out pre-training phase on sceneflow for every model

However due to the use of pre-trained weights for the RAFT network, bias is introduced since the model has likely seen all samples in the dataset. This was again done due to lack of resources and time leaving us unable to fully re-train RAFT for each split of the dataset.

With this context, we can't fully conclude that our model improves on the baseline, but we can confidently say that it has real potential to.

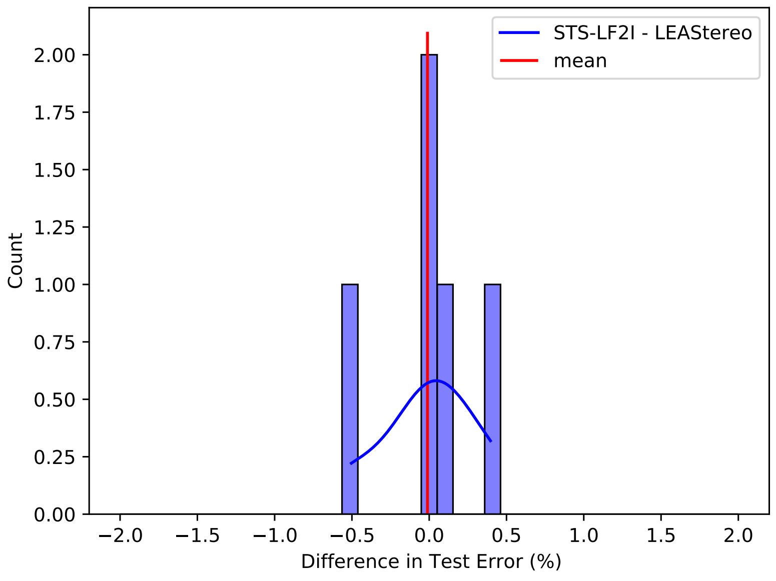


Figure 4.2: Histogram of Mean Error differences between STS-LF2I and LEAStereo across splits

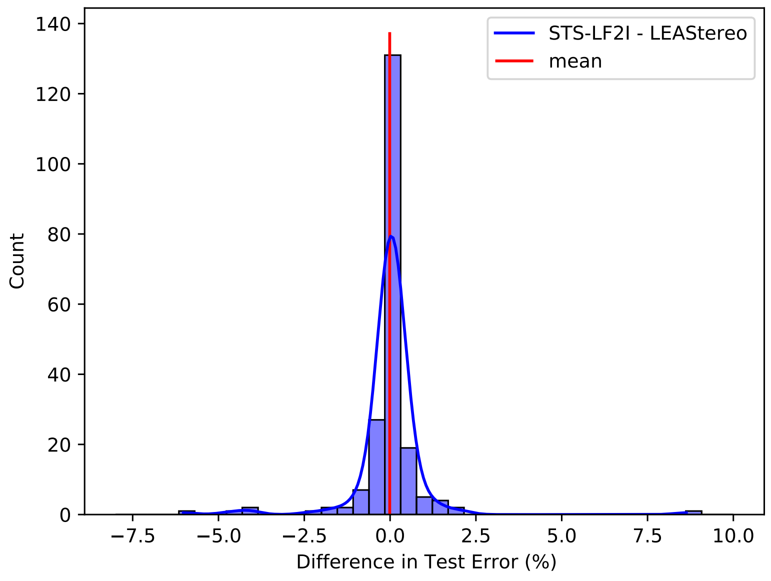


Figure 4.3: Histogram of Sample Error differences between STS-LF2I and LEAStereo across individual samples

4.4.1.2 STS-LF2

STS-LF2 on the other hand clearly performs worse than the baseline. This is likely due to additional sparsity introduced by the inversion algorithm, which is then further made worse by the warping step. STS-LF2I avoids this problem in part by generating smooth inverse Optical Flow directly.

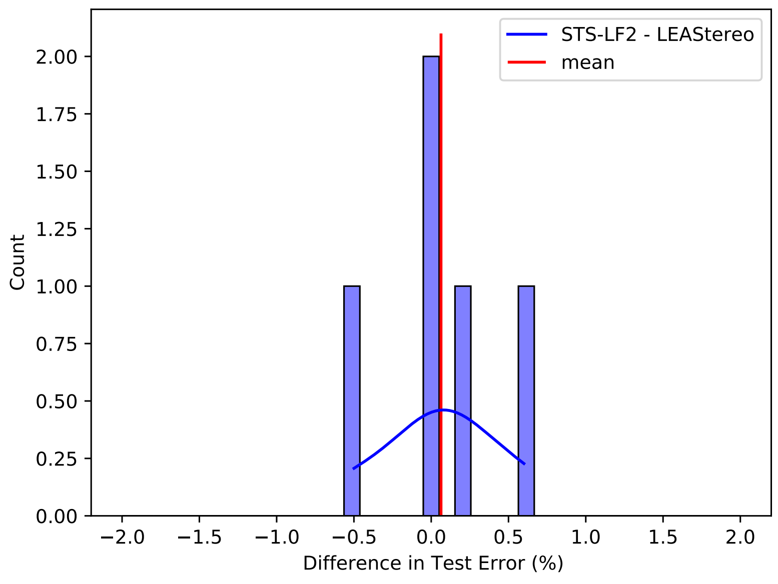


Figure 4.4: Histogram of Mean Error differences between STS-LF2 and LEAStereo across splits

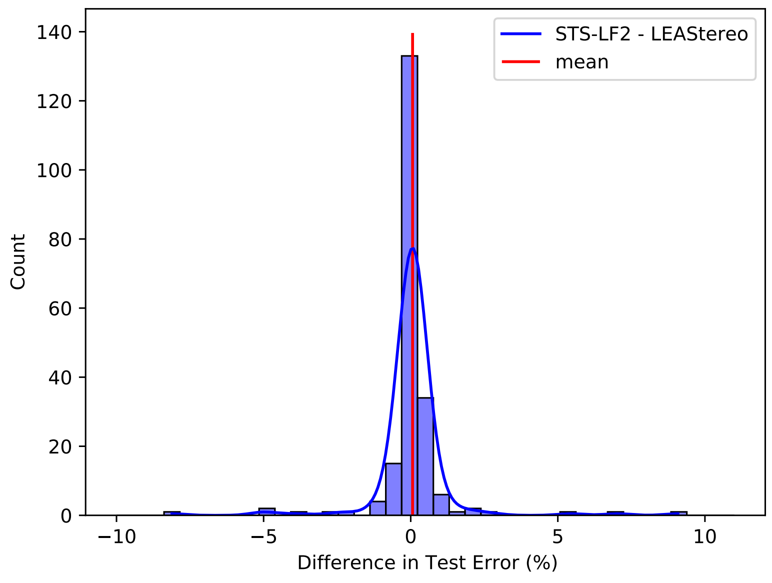


Figure 4.5: Histogram of Sample Error differences between STS-LF2 and LEAStereo across individual samples

4.4.2 Early Fusion vs. Baseline

We also show that that our best performing Early Fusion model clearly failed to perform well consistently, which indicates the original performance stemmed from a dataset split which was more optimal for the model. Only in one of the five splits, STS-EFC2B achieved lower average error than the baseline as seen in figure 4.6. The statistical test indicates LEAStereo is statistically better than STS-EFC2B (table 4.5). We also show the histograms corresponding to both tests in figures 4.6 and 4.7.

Test	Type	p-value
STS-EFC2B vs. LEAStereo	Sample	1.627e-18
STS-EFC2B vs. LEAStereo	Mean	0.125

Table 4.5: Statistical significance test results

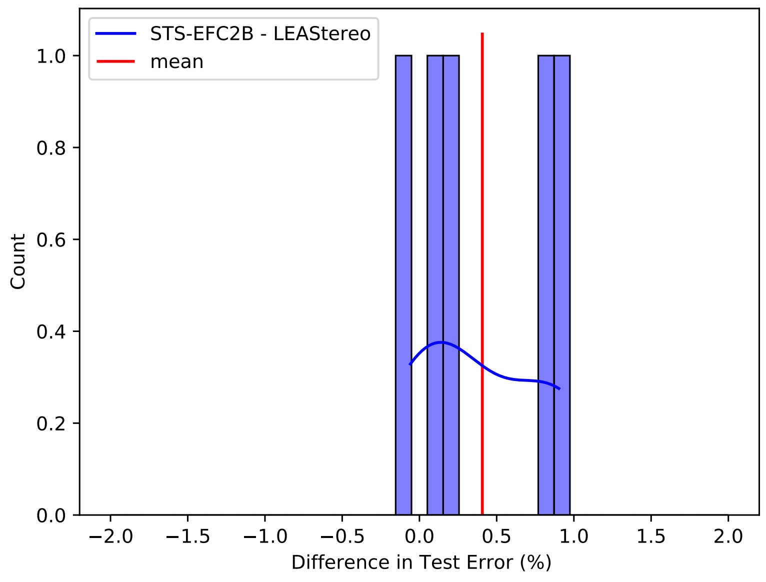


Figure 4.6: Histogram of Mean Error differences between STS-EFC2B and LEAStereo across splits

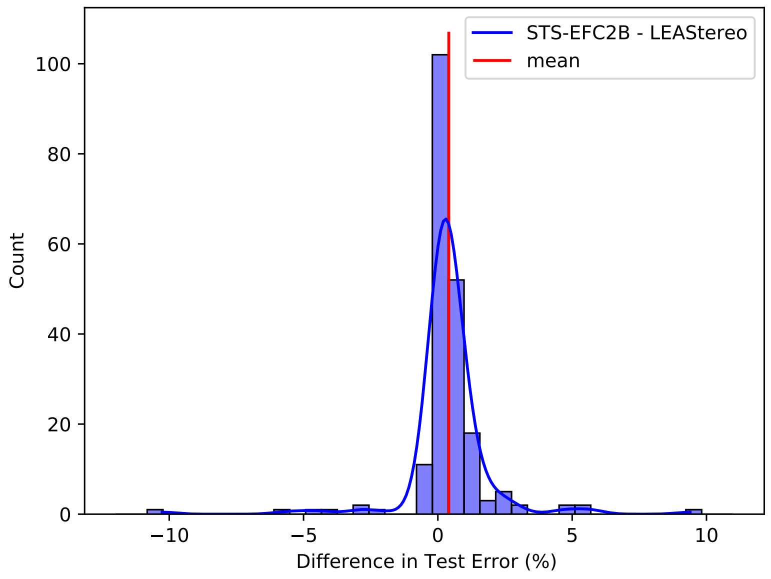


Figure 4.7: Histogram of Sample Error differences between STS-EFC2B and LEAStereo across individual samples

4.4.3 Early Fusion vs. Late Fusion

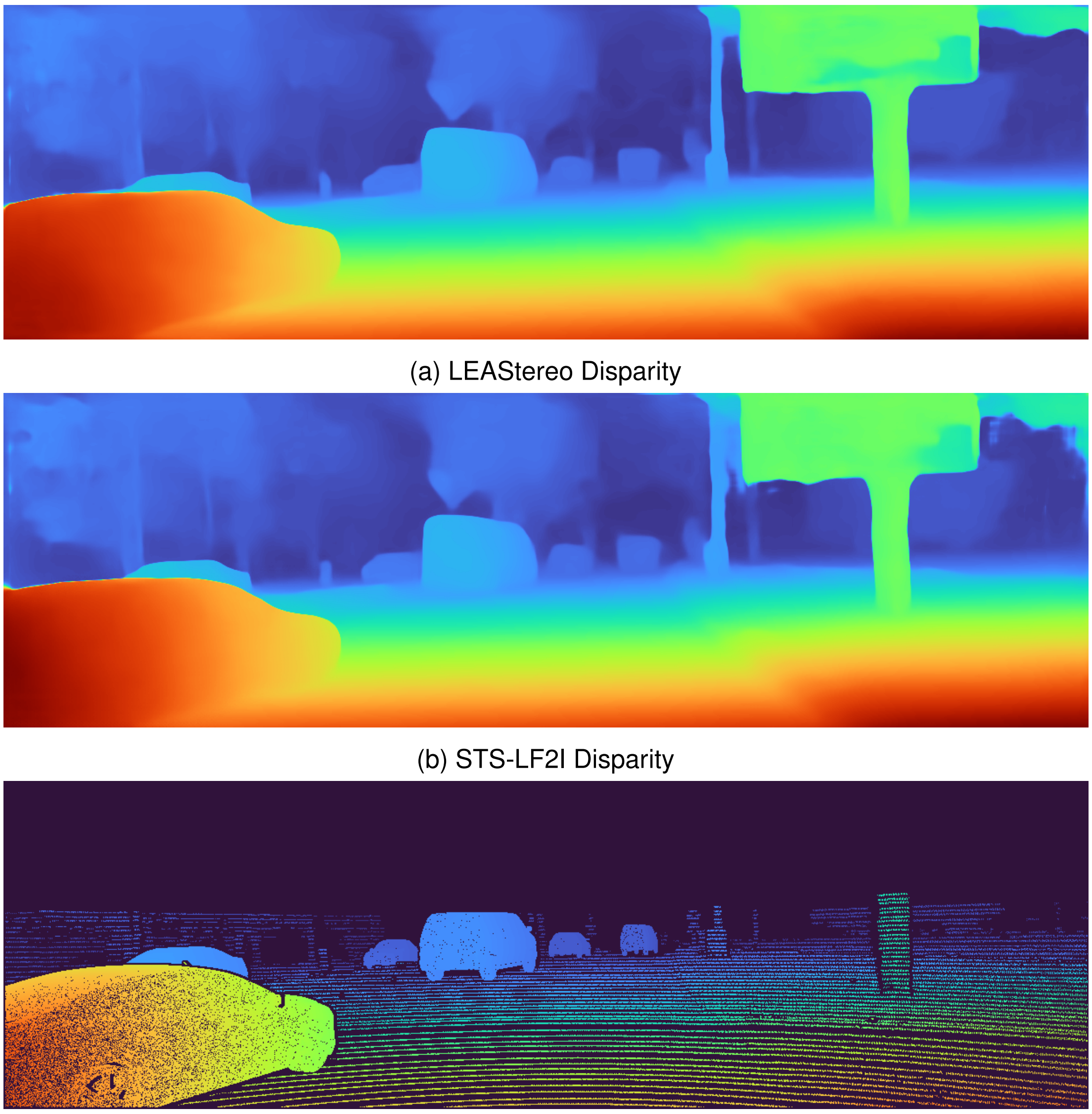
While we cannot generalize our particular Early Fusion model's performance to all Space-time Early Fusion models, we can conclude that our best Late Fusion models, performed better overall than our best Early Fusion model. It is evident that compart-mentalising the network, leads to a much cleaner architecture, which is easy to reason about. Well performing Late Fusion stereo architectures are easier to design and while less efficient, in terms of feature re-generation, they perform adequately.

In the next section we move on to analysing sources of errors in our models.

4.5 Error Analysis

We now focus on detailed analysis of the performance of STS-LF2I, our best algorithm so far. We look at sample disparity outputs, and compare error maps between our model and the baseline to identify weaknesses and strengths.

We begin by looking at an example of an image where our model achieves smaller error, and the gap between both models' errors is the highest across the entire dataset. The generated as well as ground truth disparities can be seen in figure 4.8


(a) LEAStereo Disparity

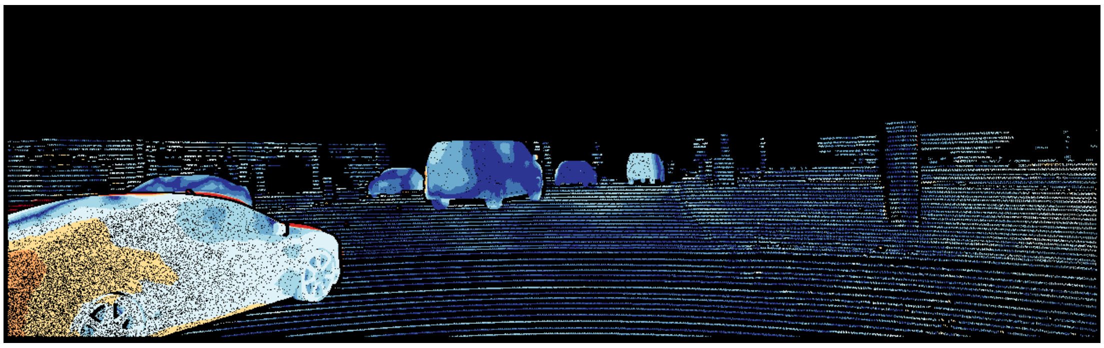
(b) STS-LF2I Disparity

(c) Ground Truth Disparity

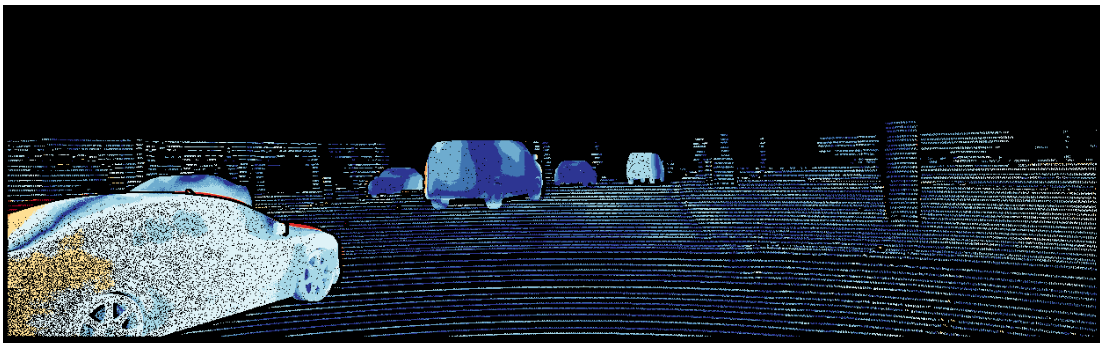
Figure 4.8: Disparities of STS-LF2I and LEAStereo for image '95' in the KITTI dataset

We also show error maps, coloured according to the logarithmic scale introduced in Menze et al. (2015). We show both individual error maps, but also subtracted error maps with negative values clipped to zero to show the areas of the image where each model performed worse. These maps can be seen in figure 4.9

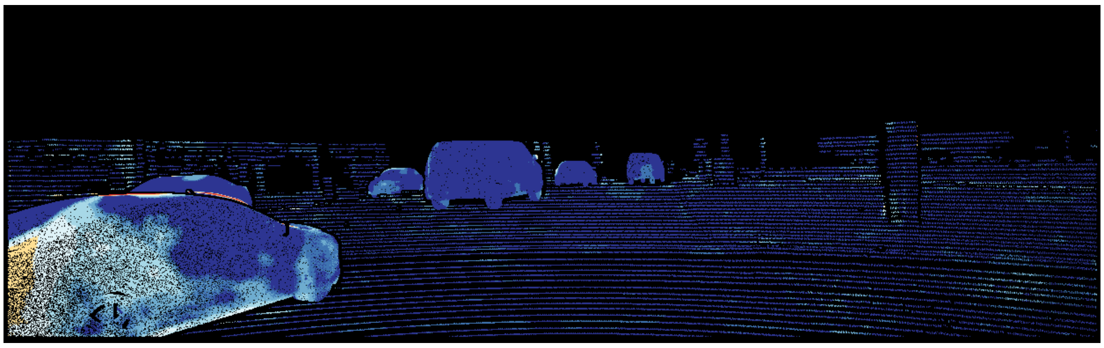
The yellow and red areas show most significant errors and differences between the models. Most notably LEAStereo produces much higher error around the edges of the car appearing at the bottom left corner of the image, and also its rear. STS-LF2I on the other hand exhibits higher error in the furthest edge of the rear of the same car.



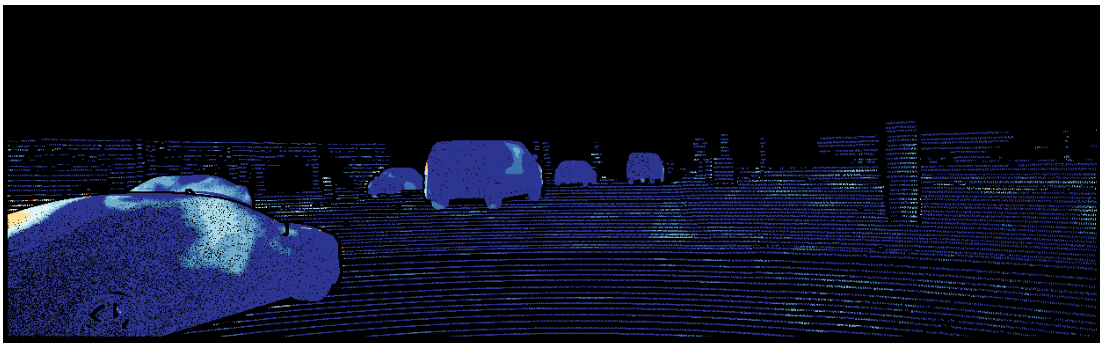
(a) LEAStereo Error Map



(b) STS-LF2I Error Map



(c) Heatmap showing areas where LEAStereo exhibited higher errors



(d) Heatmap showing areas where STS-LF2I exhibited higher errors

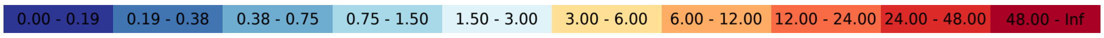
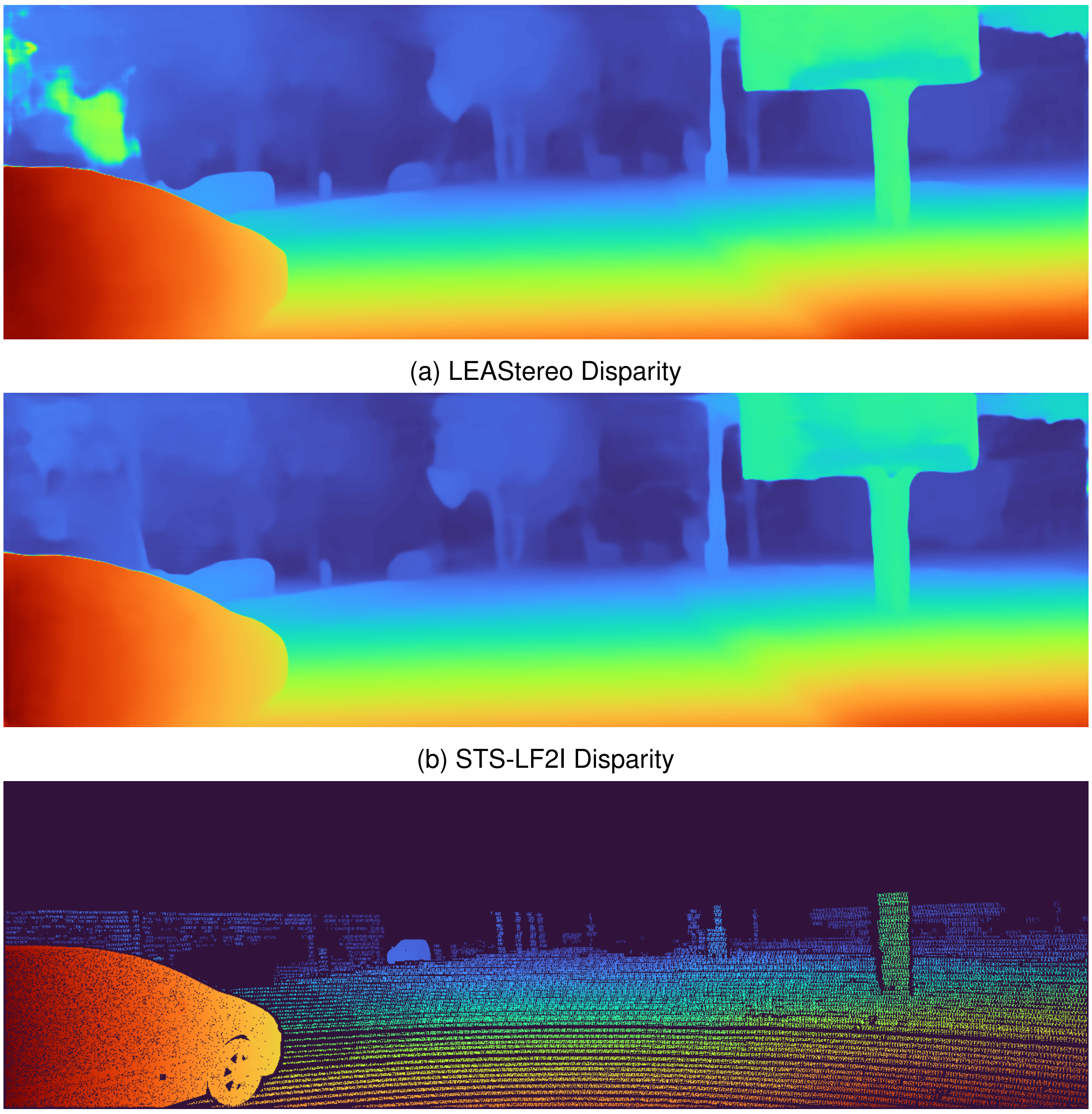


Figure 4.9: Error maps of STS-LF2I and LEAStereo for image ’95’ in the KITTI dataset. Blue-white areas are considered correct classifications, and represent values under 3px

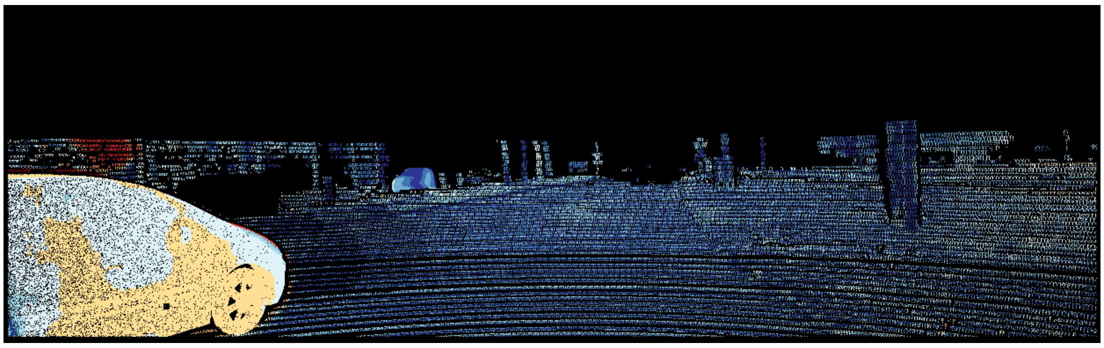
We then find an example in which our model performs worse, and where the gap is as large as possible again, and generate error maps the same way. These can be seen in figures 4.10 and 4.11

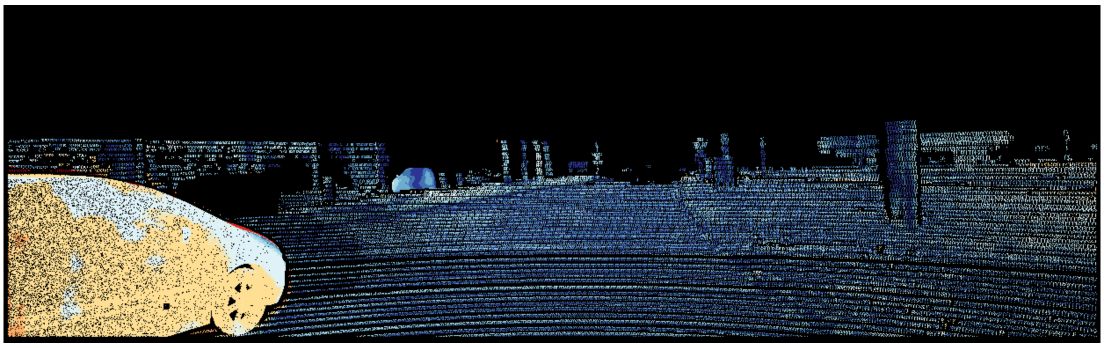

(a) LEAStereo Disparity

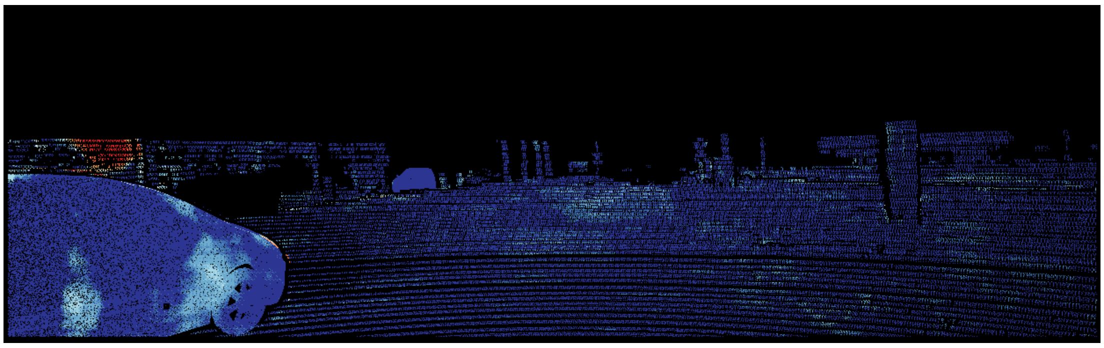
(b) STS-LF2I Disparity

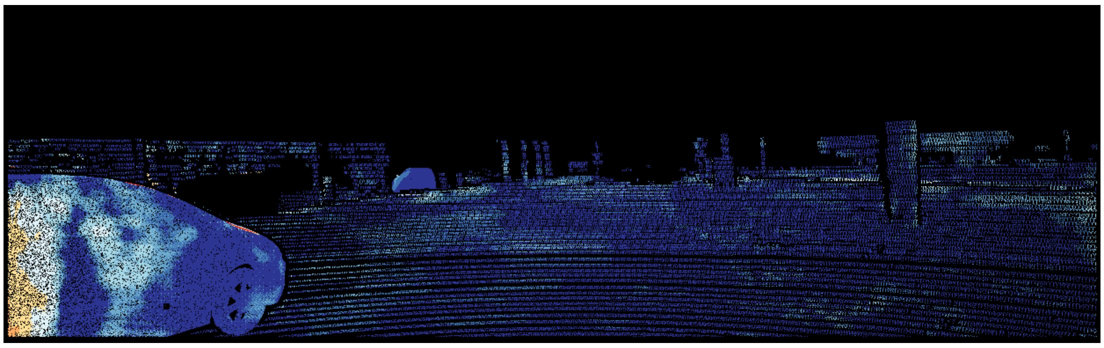
(c) Ground Truth Disparity

Figure 4.10: Disparities of STS-LF2I and LEAStereo for image '91' in the KITTI dataset, coloured according to logarithmic colour map


(a) LEAStereo Error Map


(b) STS-LF2I Error Map


(c) Heatmap showing areas where LEAStereo exhibited higher errors


(d) Heatmap showing areas where STS-LF2I exhibited higher errors

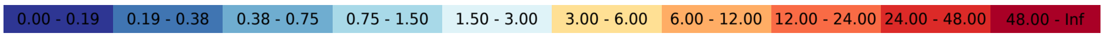


Figure 4.11: Error maps of STS-LF2I and LEAStereo for image '91' in the KITTI dataset, coloured according to logarithmic colour map

Those examples are extremely interesting, since they portray the same exact environment, with seemingly very few foreground differences. The highest contrast in performance between the methods, is found in almost identical inputs. This suggests that the improved performance in our model is due to randomness in the optimization procedure rather than genuine improvement.

Our model performs much better on the boundary around the car in figure 4.9, and yet much worse on the boundary of a similar car in the exact same position, within the same environment in figure 4.11.

These findings can be explained if one of the following is true:

- Our model improves on LEAStereo but inconsistently
- Our model does not considerably improve on LEAStereo

The second hypothesis is strongly supported by the statistical tests in section 4.4.1, and is consistent with our findings.

The first hypothesis, while not strongly supported by statistical tests, could be explained if our model benefited from the auto-encoder occasionally, perhaps by learning to refine the disparity maps instead of unifying the two time frames. This would be interesting, since the results from our STS-LFGTF model, suggest the model is in fact able to make use of features from the second time step in generating the disparity map for the first.

Since both networks only vary in one strand of the network, this change in behaviour could be explained by a combination of the following factors:

- STS-LFGTF not having to warp the image, and hence producing complete input which provides the network with more information
- Imperfect Optical Flow from RAFT could be making the inter-frame matching more prone to error. Our algorithm could be very sensitive to this, due to lack of mechanism for picking matches based on some neighbourhood around the pixel pointed to by Optical Flow.

Based on our findings here, we can pave way for future improvements to our best model. With current evidence it is difficult to rule out any of these aforementioned possibilities. With p-value under 0.1 seen in experiments in section 4.4.1, our slight improvement over the baseline could be explained through random chance. Though the probability of this is not extremely high; It is also not low enough for us to be confident in the superiority of our model.

4.6 Summary

In this section we detailed our experiments, and analysed their results. We used each series of experiments to filter the pool of all our models down to one final model.

Using our data we were able to identify STS-LF2I as the most successful prototype, and reasoned about modifications which could lead to further improvements. This prototype achieved slightly lower error than LEAStereo over five-fold cross-validation, although this was shown not to be statistically significant.

We note that overall we have trained over 52 different networks, as part of our design and experimentation stages. Some of these were faulty, and some simply would not have contributed much to our discussion. Hence we chose not to include all of our results for brevity. Each of our training runs required between 20 and 48 hours of GPU time, depending on the network being trained - meaning all of our training took 43 days as a lower bound. This is why training was not carried out in full as in original work, as this would have required at least double the time.

We conclude with a summary of the entire project and propose a direction for future work in this area.

Chapter 5

Conclusion

5.1 Summary

In this project, we explored 8 different architectures with the aim of unifying spatial and temporal features found in stereo video, in order to improve the performance of the LEAStereo network.

We organised our architectures into two main types: Late and Early Fusion, based on their architectural properties, and shaped our discussion around this split.

We then trained each of these architectures individually, in order to get an idea of their ballpark performance, and then established the possible variance in performance due to random seed in two of the models.

Through our STS-LFGTF model, we established that using an auto-encoder was an adequate mechanism for unifying two disparity outputs in the same frame of reference, and extracting information from the second frame in order to improve performance on the first.

We used these findings to pick 4 final models which we then put through 5-fold cross validation. We then run Wilcoxon statistical tests to establish the role that random chance played in the comparison.

We found that our Late Fusion models performed better than our Early Fusion models, and also identified an architecture which slightly outperformed LEAStereo, by achieving 0.012% smaller average error over all of our cross validation test splits. We found that the probability of this happening given that our model performed identically to LEAStereo was 0.094, which was not statistically significant. We analysed our model further by looking at error maps of both: our model's and LEAStereo's, on inputs over which the models displayed the biggest difference in performance. With our previous findings, we concluded that improvements could have either been explained with randomness in optimization and sampling; Or poorer performance of the model due to high sensitivity to lower quality of Optical Flow.

Our work has provided a solid foundation for future work in deep Space-time Stereo, and an insight into the characteristics of Early and Late Fusion stereo models. We hope this motivates further research, and helps push the SOTA performance even further in the future.

5.2 Future Work

Our work has identified STS-LF2I as a promising candidate for deep Space-time Stereo. Future improvements to the model should alleviate the sensitivity of the model to poor Optical Flow. We believe this could be done by enabling the network to further choose which pixels in the warped disparity from the second frame, should be matched with which pixels in the disparity of the first frame. This could be done either: Globally, for example by learning further transformations on the warped disparity before it is put through the auto-encoder; Or locally by replacing the auto-encoder entirely and learning to select temporal matches based on small neighbourhoods around the pixels.

Our project was quite limited by hardware, namely the inability to use identical batch sizes, and follow an identical training regime for all models. Hence our findings should be verified, by training the networks from scratch with sceneflow pre-training phases, at higher resolution and with larger batch sizes, like how this was done in Cheng et al. (2020). This would also check that our best model can be trained to SOTA standard like LEAStereo.

Bibliography

Bertozzi, Massimo, Alberto Broggi, and Alessandra Fascioli (2000). “Vision-based intelligent vehicles: State of the art and perspectives”. In: *Robotics and Autonomous Systems* 32.1, pp. 1–16. ISSN: 0921-8890. DOI: https://doi.org/10.1016/S0921-8890(99)00125-6. URL: https://www.sciencedirect.com/science/article/pii/S0921889099001256.

Bleyer, Michael, Christoph Rhemann, and Carsten Rother (Jan. 2011). “PatchMatch Stereo - Stereo Matching with Slanted Support Windows”. In: *BMVC*. URL: https://www.microsoft.com/en-us/research/publication/patchmatch-stereo-stereo-matching-with-slanted-support-windows/.

Chang, Jia-Ren and Yong-Sheng Chen (2018). *Pyramid Stereo Matching Network*. DOI: 10.48550/ARXIV.1803.08669. URL: https://arxiv.org/abs/1803.08669.

Cheng, Xuelian et al. (2020). *Hierarchical Neural Architecture Search for Deep Stereo Matching*. arXiv: 2010.13501 [cs.CV].

Davis, James, Ravi Ramamoorthi, and Szymon Rusinkiewicz (June 2003). “Spacetime Stereo: A Unifying Framework for Depth from Triangulation”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 359–366.

Dosovitskiy, Alexey et al. (Dec. 2015). “FlowNet: Learning Optical Flow With Convolutional Networks”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Girshick, Ross (2015). *Fast R-CNN*. DOI: 10.48550/ARXIV.1504.08083. URL: https://arxiv.org/abs/1504.08083.

Guo, Xiaoyang et al. (2019). *Group-wise Correlation Stereo Network*. arXiv: 1903.04025 [cs.CV].

Horn, Berthold and Brian Schunck (Aug. 1981). “Determining Optical Flow”. In: *Artificial Intelligence* 17, pp. 185–203. DOI: 10.1016/0004-3702(81)90024-2.

Ilg, Eddy et al. (July 2017). “FlowNet 2.0: Evolution of Optical Flow Estimation With Deep Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kendall, Alex et al. (2017). *End-to-End Learning of Geometry and Context for Deep Stereo Regression*. arXiv: 1703.04309 [cs.CV].

Lecun, Yann, Patrick Haffner, and Y. Bengio (Aug. 2000). “Object Recognition with Gradient-Based Learning”. In:

Mao, Yamin et al. (Oct. 2021). “UASNet: Uncertainty Adaptive Sampling Network for Deep Stereo Matching”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6311–6319.

Mayer, Nikolaus et al. (June 2016). “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation”. In: pp. 4040–4048. DOI: `10.1109/CVPR.2016.438`.

Menze, Moritz, Christian Heipke, and Andreas Geiger (2015). “Joint 3D Estimation of Vehicles and Scene Flow”. In: *ISPRS Workshop on Image Sequence Analysis (ISA)*.

Molton, N. et al. (1998). “A stereo vision-based aid for the visually impaired”. In: *Image and Vision Computing* 16.4. Vision-Based Aids for the Disabled, pp. 251–263. ISSN: 0262-8856. DOI: `https://doi.org/10.1016/S0262-8856(97)00087-5`. URL: `https://www.sciencedirect.com/science/article/pii/S0262885697000875`.

Nover, Harris, Supreeth Achar, and Dan B Goldman (2018). “ESPReSSo: Efficient Slanted PatchMatch for Real-time Spacetime Stereo”. In: *In Proceedings of Sixth International Conference on 3D Vision (3DV)*.

Scharstein, Daniel and Richard Szeliski (Apr. 2002). “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms”. In: *International Journal of Computer Vision* 47.1, pp. 7–42. ISSN: 1573-1405. DOI: `10.1023/A:1014573219977`. URL: `https://doi.org/10.1023/A:1014573219977`.

Sun, Deqing et al. (June 2018). “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tian, Liang et al. (Oct. 2019). “Disparity estimation in stereo video sequence with adaptive spatiotemporally consistent constraints”. In: *The Visual Computer* 35.10, pp. 1427–1446. ISSN: 1432-2315. DOI: `10.1007/s00371-018-01622-1`. URL: `https://doi.org/10.1007/s00371-018-01622-1`.

Tsekourakis, Iraklis and Philippos Mordohai (June 2019). “Measuring the Effects of Temporal Coherence in Depth Estimation for Dynamic Scenes”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2867–2875. DOI: `10.1109/CVPRW.2019.00346`.

Zachary, Teed and Deng Jia (2020). “RAFT: Recurrent All-Pairs Field Transforms for Optical Flow”. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Cham: Springer International Publishing, pp. 402–419. ISBN: 978-3-030-58536-5.

Zbontar, Jure and Yann LeCun (June 2015). “Computing the Stereo Matching Cost With a Convolutional Neural Network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, Feihu et al. (2019). *GA-Net: Guided Aggregation Net for End-to-end Stereo Matching*. arXiv: `1904.06587 [cs.CV]`.

Zhang, Li, B. Curless, and S.M. Seitz (2003). “Spacetime stereo: shape recovery for dynamic scenes”. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2, pp. II–367. DOI: `10.1109/CVPR.2003.1211492`.

Zhang, Li et al. (2004). “Spacetime faces: high resolution capture for modeling and animation”. In: *ACM SIGGRAPH 2004 Papers*.