Making It Clear

Using Vision Transformers in Multi-View Stereo on Specular and Transparent Materials

W.E.P. Tolsma

TU Delft

FIZYR

Making It Clear

Using Vision Transformers in Multi-View Stereo on Specular and Transparent Materials

Thesis report

by

W.E.P. Tolsma

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on October, 11th 2023 at 14:00

Thesis committee:

Chair:	Assoc. Prof. Dr. J.C. van Gemert
Supervisors:	Assist. Prof. Dr. N. Tömen
External examiner:	Assist. Prof. Dr. M. Weinmann
Project Duration:	February 2023 - October 2023
Student number:	4531124

Faculty of Electrical Engineering, Mathematics and Computer Science · Delft University of Technology

Preface

This report marks the end of my time as a student at the Delft University of Technology. Pursuing a Master's degree in Artificial Intelligence during the 'AI boom' was interesting, to say the least. It was a humbling experience to try and keep up with the rapid pace at which related research papers were released, which says a lot about the amount of attention this field currently has by the community. Hopefully, this report will give you a glimpse of the exciting possibilities we currently already have in applying AI for challenging tasks.

I wish to give a special thanks to Nergis Tömen, who committed to meeting with me during her personal time off. I want to thank Ronald Poelman, for inspiring and guiding me throughout the discovery of my thesis, and for providing an endless supply of unread papers. I want to thank Hesam Araghi, for always being available to listen to my ramblings and giving timely feedback. I also want to thank Jan van Gemert, for making this project possible at the PRB group and providing guidance on where to take this project.

Finally, I want to thank my girlfriend, my friends, and my family for their support during the past year.

Pieter Tolsma

Delft, September 2023.

Contents

1 Introduction

Depth estimation is a cornerstone of modern perception systems, powering a wide range of high-tech products that enable things like self-driving, augmented reality, and warehouse automation. An accurate depth estimate helps these systems reason about and act on their environment in a more advanced way than a simple 2D image allows. Multi-View Stereo (MVS) combines multiple images from different viewing angles in order to predict depth, which one could argue has similarities to how we humans perceive the world. Compared to active sensing, which requires specialized hardware and has a dependency on the specific environment, MVS has the potential to be a cheap, scalable, and widely applicable alternative. Sadly, these methods are plagued by ambiguities like occlusion, texture-less regions and surfaces with strong view-dependent effects, such as reflections and transparencies. Recent advancements in machine learning have allowed MVS to book steady progress over the past few years in this area.

A relatively recent discovery is the Transformer [1] architecture, which has shown significant potential in solving complex problems. The Vision Transformer (ViT) [2] enabled researchers to start using transformers in vision, which has sparked a new area of research. Compared to convolutional neural networks (CNNs), the ViT is said to be more shape-focused, less biased towards texture, and supposed to be better at transfer learning [3][4][5]. In the area of MVS, recent works have also started using ViTs for estimating depth [6]. If the ViT is really more robust and is more shape-focused, does this hold promises for MVS on specular and transparent materials? Based on the 2D conceptions of the ViT, we can only hypothesize whether this is true, and it is difficult to know without a proper dataset to evaluate this.

In this thesis, we put the ViT to the test in the domain of MVS by comparing it to a proven CNN alternative. We evaluate the models by generating a unique, multi-material MVS dataset that allows for the controlled evaluation of diffuse, specular and transparent materials. To the best of our knowledge, there is no other dataset that allowed us to do this.

The structure of this report is as follows. Chapter 2 contains the main article, which is written with the assumption that the reader has sufficient background knowledge in computer vision. Chapter 3 contains background information, which can help the reader understand the work in the previous chapter. In this chapter, we explain the basics of depth estimation, look at simple to advanced deep-learning models, and expand on the various data sets that are publicly available.

2

Scientific Article

Page intentionally left blank.

Making it Clear: Using Vision Transformers in Multi-View Stereo on Specular and Transparent Materials

W.E.P. Tolsma
Delft University of Technology
epieter.tolsma@gmail.com

H. Araghi
Delft University of Technology
H.Araghi@tudelft.nl

N. Tömen
Delft University of Technology
N.Tomen@tudelft.nl

R. Poelman
Fizyr B.V.
R.Poelman@fizyr.com

J.C. van Gemert
Delft University of Technology
J.C.vanGemert@tudelft.nl

Abstract

Transparency and specularity are challenging phenomena that modern depth perception systems have to deal with in order to be used in practice. A promising family of depth estimation methods is Multi-View Stereo (MVS), which combine multiple RGB images to predict depth, thus circumventing the need for costly specialized hardware. Although promising, finding pixel-to-pixel mappings between images is a challenging task, clouded by ambiguity. In order to determine the current ability to deal with such ambiguity, we introduce ToteMVS: a multi-view, multi-material synthetic dataset with diffuse, specular and transparent objects. Recent works in computer vision have effectively replaced Convolutional Neural Networks (CNNs) with the emerging Vision Transformer (ViT) architecture, but it remains unclear whether ViTs outperform CNNs in handling reflective and transparent materials. In our study, we use ToteMVS to compare ViT- and CNN-based architectures on the ability to extract useful features for depth estimation on diffuse, specular, and transparent objects. Our results show that, in contrast with the current trend of using ViTs over CNNs, the ViT-based model does not have a special capability for dealing with these challenging materials in the context of MVS. Our evaluation data, including related code, can be found on our GitHub.

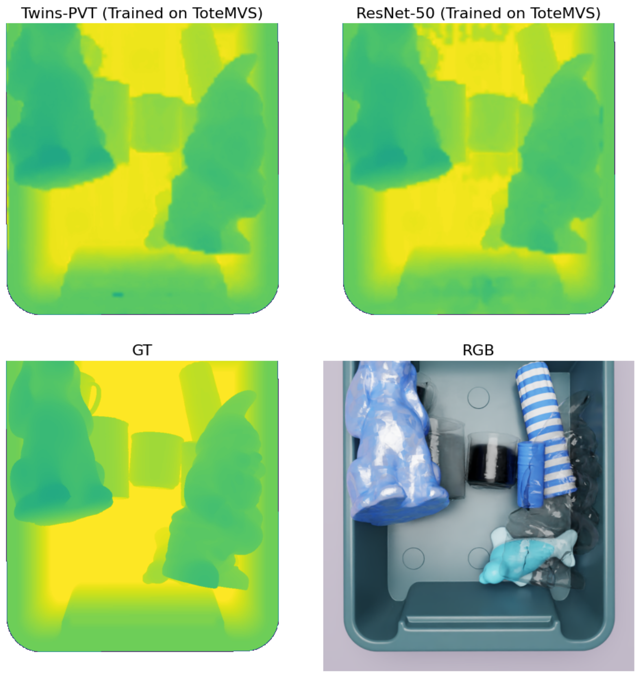

Twins-PVT (Trained on ToteMVS)
ResNet-50 (Trained on ToteMVS)
GT
RGB

Figure 1: Estimating depth using six cameras on diffuse, specular and transparent materials using the MVSFormer[4] model, with either a Twins-PVT (a ViT) [7] or a ResNet-50[14] (a CNN) in the backbone. Both variants are able to extract the almost invisible dwarf with high accuracy once trained on our data. This figure highlights the high output similarity between the two approaches.

1. Introduction

Estimating depth on transparent objects is a challenging task in computer vision partly due to the ambiguity of the background appearing through the foreground (Fig. 1). Specularities are another nuisance that modern vision systems have to deal with in order to be robust to varying light intensities and directions. Multi-view stereo (MVS) uses multiple RGB images to predict depth through epipolar geometry, a common and versatile method for estimating depth. Recently, Vision Transformers (ViTs) have started to appear in MVS architectures, but their effectiveness is still largely undiscovered in this domain. In this work, we analyze the capability of state-of-the-art MVS on the difficult task of estimating depth on specular and transparent objects and answer the question of whether a modern ViT can be a superior backbone for this compared to a Convolutional Neural Network (CNN). To answer this question, we introduce ToteMVS: a multi-material synthetic MVS dataset that allows the evaluation of models on diffuse, specular, and transparent materials in a controlled setting.

Our motivation behind scrutinizing modern MVS on these materials is fueled by the imaginative factor that is inherent to any deep-learning-based system. This property could make these systems robust to ambiguous problems that specular and transparent objects create. Obtaining such a system would be an engineering feat, as current off-the-shelf sensors fail to address these situations properly. For example, sensors relying on structured light or time-of-flight [55] are based on the reflection of emitted signals on surfaces, a property that only works consistently when the surface is diffuse (reflects light in all directions). MVS does not require specialized hardware, nor does it interact with the environment, making it a candidate for a general, all-purpose depth estimator.

Despite these promising qualities, current MVS methodologies are plagued by the ill-posed nature of the problem it is solving, which at its core can be explained by finding pixel correspondences between images. When the search space is mostly untextured, finding a match leads to too many possibilities. When a surface point is occluded, it is possible that a match can not be found [52]. Lastly, when the appearance of surfaces changes, for example, due to reflections or transparencies, finding a match becomes significantly more difficult[37].

Fortunately, in the past decade, the field of pattern recognition in computer vision has experienced a large number of developments [3], which has enabled MVS to become more accurate on benchmarks like DTU [2] or Tanks&Temples [18]. While these benchmarks are an important measuring stick for tracking overall progress, they do not allow for the controlled evaluation of different materials, which can theoretically make a large difference. Measuring this difference becomes important for deciding on the effectiveness of architectural choices.

Modern MVS can generally be deconstructed into four stages: feature extraction, warping, regularization, and refinement (see Fig. 10). Currently, the two main categories for feature extraction are Convolutional Neural Networks (CNNs) [21, 20] or Vision Transformers (ViTs)[19], the latter being a more recent discovery with unique properties, such as the immediate global understanding of the input, compared to the CNN gradually *seeing* more of the image. No heuristic currently exists that can tell whether a ViT should be preferred over a CNN, making empirical tests the only practical option. There is very little research on whether ViTs can push the state-of-the-art in MVS when dealing with challenging materials, while literature suggests that ViTs are more robust and shape-focused [30, 12, 33]. This could potentially make the ViT a superior choice when dealing with specular and transparent materials, which suffer strong view-dependent effects. The state-of-the-art method MVSFormer [4] includes a ViT in the feature extraction process and reports improvements compared to a ResNet-50 [14], making it a prime candidate architecture for further evaluation. In order to eliminate potential model-specific dependencies on the extracted features, the same experiments are run on IterMVS[41], a different method that is modifiable in the exact same way.

In this work, we compare the Twins-PVT [7] architecture to a ResNet-50 [14], to see whether a ViT is more effective in extracting features on specular and transparent objects for MVS. We find that the Twins-PVT architecture generalizes significantly better when trained on DTU to our data compared to the ResNet-50, but notice no specific advantage on transparent materials. Once we finetune both models on our data, we find that both the ViT and CNN have similar capabilities in extracting features on specular and transparent materials. When textural features are removed, we do not observe any extra robustness for the ViT, which is surprising as literature suggests that CNNs are generally more dependent on texture compared to ViTs. Our conclusions are an insight into the mild effectiveness of a ViT versus a CNN, going against the trend of simply replacing a CNN with a ViT in order to significantly boost performance.

Our main contributions can be summarized as follows:

- We introduce ToteMVS: a novel multi-material synthetic MVS dataset that allows for comparison between diffuse, specular, and transparent materials in a controlled manner. We show that transparent objects are significantly more challenging than their diffuse and specular counterparts.

- We provide a quantitative and qualitative comparison between Twins-PVT and ResNet-50 backbones in MVS on the three material types. We find that both backbones achieve similar results, but that the ViT-based model has significantly better generalization performance.

2. Related work

Depth Perception Systems Structure from Motion (SfM) [39][36] is a popular method for computing depth and camera positions based on a large number of images in a scene. By matching features with high confidence, SfM typically produces sparse yet accurate point clouds, making it a useful initial calibration method for further processing. Active depth sensing, such as time-of-flight (ToF) and structured light [55], predict depth by probing the environment with signals. This way of predicting depth has the potential to be accurate but requires specialized hardware, depends on environmental conditions, and may fail on challenging surfaces such as reflective or transparent. Neural Radiance Fields (NeRFs) [28], provide a novel way of combining multiple images in order to render novel views, and are also capable of estimating depth. While NeRFs excel at novel view synthesis, they fall short on generalization, efficiency, and the ability to produce accurate depth maps with sparse-view image sets. Stereo vision [53] mimics the human visual system by combining two or more cameras to estimate depth. In essence, this method uses the parallax effect and an estimation of the disparity between images in order to predict depth. This relatively simple method faces challenges with texture-less regions, occlusions, and view-dependent effects such as reflections or transparencies. Recent works have built upon the concept of stereo vision with promising results in these areas, by combining recent deep-learning advancements with classic epipolar geometry.

Multi-View Stereo with Deep Learning MVS methods combine views by exploiting classic epipolar geometry [11, 51]. The core of the MVS problem can be described as finding a pixel-to-pixel mapping between a pair of images. This problem is sometimes described as ill-defined, as there are situations where it is impossible to determine a matching point, caused by effects such as textureless regions or strong view-dependent effects. Texture-less regions make the search for a corresponding point difficult, as there exist too many candidates to choose from. View-dependent effects, such as reflectiveness or transparency, make surfaces appear different between camera angles, making it difficult to find correspondence based on color. Transparency has the additional problem that the background appears through the foreground, adding another layer of obfuscation. Recent works [15, 49, 13, 42, 41, 45, 4] use advanced feature extraction backbones, combined with novel fusion strategies. A typical pipeline of these advanced MVS methods is depicted in Fig. 2, a design that was first used by the authors of MVSNet [49, 44]. A detail that stands out is the fact that all of these methods use a Feature Pyramid Network (FPN) [22], which is designed in the shape of an hourglass with skip connections. Dense prediction tasks, like per-pixel depth estimation, often require feature outputs that have the same resolution as the original input, for which the FPN is designed. PSMNet [5] was one of the first MVS methods to start using these multi-scale features. The multiple layers allow for a coarse-to-fine approach, where initial depth estimations are made using a low-resolution, but a high-level understanding of the scene, gradually scaling up until the full resolution is used. MVSFormer [4] *enriches* the FPN with features extracted through a ViT, which is a modification that can be applied to any method that uses an FPN, making it a design worth investigating. A simplified diagram of the MVSFormer architecture is depicted in Fig. 3.

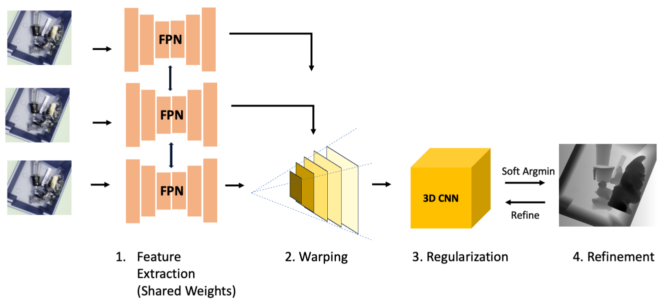


Figure 2: Typical pipeline for deep-learning based MVS. Feature extraction transforms the RGB space into a high-dimensional understanding of the original input. Warping maps the feature vectors from the source images into the frustum of the reference view, at fixed depth intervals. Regularization propagates local information, thereby smoothening the final prediction. Refinement iteratively predicts depths from a coarse to fine scale.

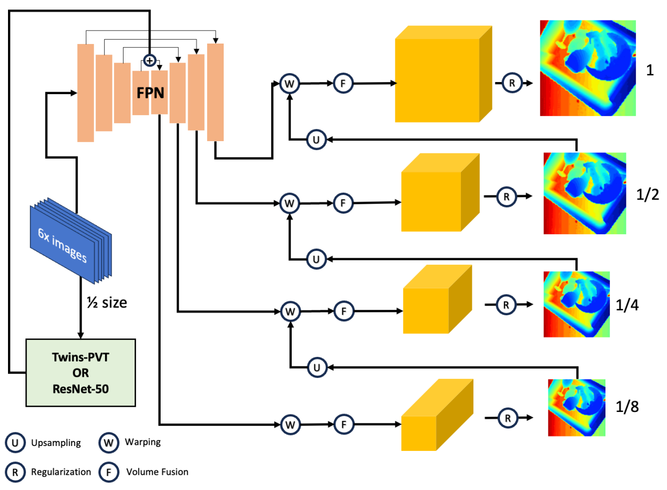


Figure 3: Simplified visualization of the MVSFormer [4] architecture. Note how the FPN is *enriched* with additional features extracted from the Twins-PVT [7] (ViT) or a ResNet-50 [14] (CNN), a technique that can be generally applied to any method using an FPN.

Depth Estimation on Transparent Materials Estimating depth on transparent materials is a task that previous works have tried to solve. MVTrans [46] proposed a method following the flow of Fig. 2 and reported success with segmentation, but demonstrated limitations with estimating depth on clear objects. Another work fused raw depth-sensor output with RGB (RGB-D) [54] which resulted in somewhat accurate yet incomplete predictions on transparent objects. TODE-Trans [16] combines RGB-D with a Vision Transformer (ViT) [19] which they claim significantly improves the completeness of the output compared to a CNN.

CNNs vs ViTs The Vision Transformer (ViT) architecture [19] has a fundamentally different approach to CNNs in processing images and has been demonstrated to be competitive with CNNs. The initial ViT implementation was unable to beat similarly sized CNNs [47], but nevertheless served as a good demonstration of the multi-modal capability of the transformer architecture[40]. Recent adaptations of the ViT [47][24] [7] introduced convolutional operations that reduced the computational complexity of self-attention, making it a viable alternative to popular backbones such as the ResNet[14]. Twins-PVT [7], used by MVSFormer [4], is an improvement to the original pyramid vision transformer (PVT) [43]. These pyramid-like transformers are suitable for dense prediction tasks, which typically require feature maps at various resolutions for a more high-level understanding of the input. Twins-PVT proposes locally grouped self-attention (LSA) and global sub-sampled attention (GSA), which enable efficient self-attention such that it can be used in real-time vision backbones. LSA is first computed over parts of the image, after which information is globally exchanged using GSA.

Some of the drawbacks of using ViTs over CNNs are the lack of translation invariance, the need for more data for effective use, and the increase in computational complexity. CNNs can also deal with varying sizes in input resolution, whereas the ViT requires multi-scale training to be able to deal with this [4]. As suggested by [32], ViTs have the potential to learn more robust feature representations, increasing the generalization capability compared to a CNN. As shown by [34], ViTs seem to produce more similar feature representations throughout the network, compared to a ResNet model having a much lower similarity between the lower and higher layers. The authors also find that the final feature outputs of the ViT have quite different representations than a ResNet output. Because of the self-attention mechanism, ViTs can 'see' the full image from the start, compared to CNN which starts with a small receptive field that increases with the network depth. TODE-Trans [16] reported a significant increase in performance metrics by using a ViT over a CNN. The authors hypothesize that it could be explained by global understanding caused by the self-attention. As shown by [12], CNNs tend to focus decisions on texture, whereas ViTs are more robust to textural differences by focusing more on shape [30]. This could potentially have consequences in MVS, where a change in texture appearance can be caused by moving the camera around.

Compared to 2D tasks, ViTs have only just started being used in MVS [4] [45] [10], and while impressive numbers are being reported, we are lacking more fine-grained insight into the characteristics of the CNN compared to a ViT, as we have for the 2D cases. More specifically, we are interested in finding out if the ViT is superior in MVS in terms of generalizability, in dealing with specular and transparent objects, and when confronted with a lack of texture.

Benchmarks. Various popular benchmarks exist on monocular or stereo depth estimation networks, like KITTI [27] and SceneFlow [25]. For benchmarking MVS methods, popular benchmarks like DTU [2], Tanks&Temples [18], ETH3D and BlendedMVS [50] all fail to include more challenging scenes with transparent objects. For transparent objects, ClearGrasp [35] and Trans10K [48] are monocular datasets. TOD [23] explains how to effectively extract ground-truth depth data from real-world transparent objects, but is also monocular. ClearPose [6] is a large real-world, annotated MVS dataset with transparent objects, but does not have diffuse or specular objects to compare with, making it difficult to give a fair comparison with equal conditions. MVTrans [46] recently published Syn-TODD: a large-scale synthetic MVS dataset with transparent objects, which nearly fits the purpose of this work, but misses the focus on discrete materialistic changes on the same objects, contains varying camera baselines, and is in general not an ideal controlled setting due to high variation in background (HDRI with strongly textured flat planes).

Using synthetic over real data has drawbacks, which can be summarized as the gap between the synthetic and real data. Synthetic data may contain artifacts or patterns that a model may overfit on [38], causing suboptimal generalization to the real world. The advantages of using synthetic data are vast and only increase, as the ability to render realistic scenes efficiently improves over time. Some of these advantages are accurate ground-truth depth and camera positions, automatic labeling, ease of access to large volumes, and the flexibility to make changes to the scene compositions as desired. Using this, data can be generated that is in the long tail of improbable occurrences, which is notoriously expensive data to collect in the real world. In the case of MVS, the biggest advantage is having access to ground-truth depth data, which would otherwise be challenging and expensive to obtain.

3. Method

3.1. ToteMVS: A Novel Multi-Material MVS Dataset

Detecting differences in performance between diffuse, specular, and transparent objects requires a controlled environment, including segmentation and ground-truth depth maps. To fulfill these requirements, we generate our own

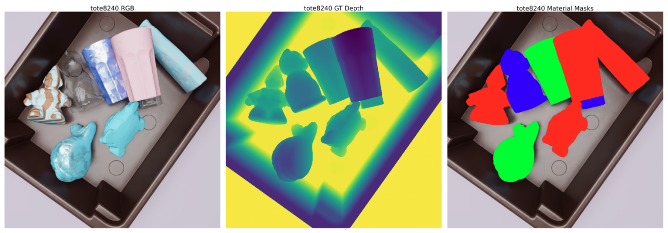


Figure 4: ToteMVS modalities. From left to right: RGB image, depth map and segmentation map. Diffuse, specular and transparent are indicated by red, blue and green respectively.

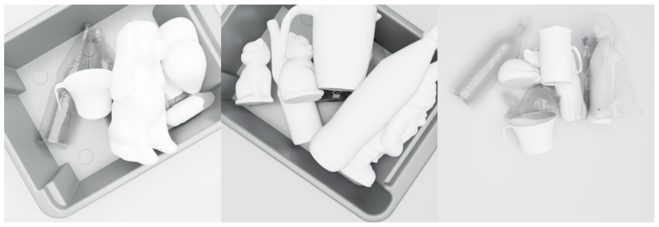


Figure 5: Textureless ToteMVS variation sample, where all texture and color is removed from the non-transparent objects. This data is used to measure the dependency on textural features.

synthetic dataset titled **ToteMVS**: a large, labeled dataset containing six camera angles, 9000 scenes with 40 different objects, and more than 1000 possible materials, including ground-truth depth data (see Fig. 4 for a sample). The resolution is 768x768, which is similar to other what common methods use for input size. Our dataset is generated using NVIDIA Isaac Sim [1], a state-of-the-art simulation environment that allows for realistic rendering of our scenes. The test set (450 scenes) can be found on our GitHub, opening up the possibility for others to evaluate models against our different material types.

Variation in ToteMVS In order to increase the generalization potential of ToteMVS, variation is introduced per scene through randomly permutating the following selected properties:

- Objects: during the simulation phase, 8 randomly selected object models are dropped from a fixed height onto the plane, which contains a randomly rotated tote bin half of the time. The 40 object models were selected from the KIT object models database [17].
- Light: besides an HDRI map that emits light, a sphere light is randomly positioned over the scene with varying intensity, color temperature and radius.
- Material: For every object, a random material class is assigned (diffuse, specular, transparent), after which texture randomization is applied using our parametrized material graph. The plane is assigned a random color in the interval $[0, 0.5]$ for every color channel. For added variation, random bump noise and roughness are added within certain ranges depending on the material.

Textureless ToteMVS In order to measure the effect of textural cues on the ability to estimate depth, a second test set is generated consisting of 450 scenes, where all texture and color are removed from the non-transparent objects (see Fig. 5).

3.2. Evaluation

For evaluating our results, we use metrics frequently used in other depth estimation works [29]. Both metrics are calculated over the set of N pixels, where every pixel $i \in N$ has a predicted depth d_i and ground-truth depth d_i^{gt}.

- Sq. Rel: $\frac{1}{N}\sum_{i\in N}\frac{|d_i - d_i^{gt}|^2}{d_i^{gt}}$
- δ-Accuracy: % of pixels s.t. $max(\frac{d_i}{d_i^{gt}}, \frac{d_i^{gt}}{d_i}) < \delta$

The squared relative error (Sq. Rel.) accounts for the fact that far-away ground truth is more likely to have a larger absolute error. The squared term weighs outliers more heavily, thus being a metric that is more sensitive to large outliers. The accuracy metrics are defined in this work as $\delta_i = 1.0125^i$ for $i \in [1, 2, 3]$ meaning the threshold is increased in three steps. This metric indicates the percentage of pixels that are within the given relative error bounds. As an example, if $\delta_1 = 0.9$, it means that 90% of the pixels has a predicted depth within 1.25% of the ground-truth depth.

3.3. Design of Experiments

Because ViTs have only recently started being used in MVS, obtaining a deeper understanding of its different capabilities is important in order to justify the architectural decision behind it. Our experiments contribute to this understanding by comparing a ViT- and CNN-based MVS backbone on diffuse, specular and transparent materials.

MVSFormer[4] is chosen for its novel enrichment of the FPN with a ViT. For the control experiment, IterMVS[41] is selected. Compared to MVSFormer, IterMVS has a different approach to depth refinement and uses a Gated Recurrent Unit (GRU). The aim of experimenting with the second method is to verify that the relationships found in the results generalize to other methods.

To evaluate the role of the ViT in predicting depth on different materials, the Twins-PVT is replaced with the

Material	Experiment	Sq.Rel.↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
Diffuse	IterMVS (ResNet-50)	0.8923	0.7864	0.8855	0.9181
	IterMVS (Twins-PVT)	0.8785	0.7944	0.8883	0.9183
	MVSFormer (ResNet-50)	0.5382	0.8730	0.9224	0.9413
	MVSFormer (Twins-PVT)	0.5568	0.8804	0.9206	0.9399
Specular	IterMVS (ResNet-50)	0.8977	0.7758	0.8831	0.9164
	IterMVS (Twins-PVT)	1.0386	0.7770	0.8776	0.9097
	MVSFormer (ResNet-50)	0.4920	0.8783	0.9282	0.9482
	MVSFormer (Twins-PVT)	0.5455	0.8825	0.9257	0.9446
Transparent	IterMVS (ResNet-50)	1.9467	0.4731	0.6899	0.7930
	IterMVS (Twins-PVT)	2.2888	0.4689	0.6763	0.7770
	MVSFormer (ResNet-50)	0.9627	0.6992	0.8525	0.9007
	MVSFormer (Twins-PVT)	0.8555	0.7159	0.8620	0.9098

Table 1: Test scores on ToteMVS after training on ToteMVS. Pretrained backbones are used. $\delta_i = 1.25^i\%$ relative error threshold.

ResNet-50[14] (see Fig. 3). The Twins-PVT (small) network has around 24.1M parameters, whereas the ResNet-50 has 26.79M, and both backbones have publicly available weights that have been trained on ImageNet-1K [9]. This is important since ViTs generally require more data to be effective compared to a CNN. Both the ViT and CNN receive input that is half the resolution that the FPN receives, an optimization suggested by MVSFormer in order to reduce computational cost.

4. Experiments

Training The experiments are run using a single NVIDIA V100 with 16GB of memory, via the Delft High Performance Computing Cluster (DHPC) [8]. After training for 36 hours, the test scores from MVSFormer stagnated around epoch 11. From the six available viewpoints, a random camera is set as the reference camera making the remaining the source cameras. The data is divided into a 95/5 train/test split, meaning 8550 scenes are used for training and the remaining for 450 testing. The original hyperparameter and learning rate scheduler configuration is copied from MVSFormer[4]. The correctness of the training setup is confirmed by reproducing the results from the publicly available weights (Table A.3).

4.1. Generalization Capability

According to [30, 33, 32], ViTs trained on ImageNet [9] generalize exceptionally well to other domains. This experiment aims to determine whether this holds for the diffuse, specular, and transparent cases in MVS. To achieve this, pre-trained (ImageNet) ResNet-50 and Twins-PVT models are configured in MVSFormer. The models are first trained on the DTU dataset, after which they are evaluated on the ToteMVS dataset. Following the claim of [30], we hypothesize that the ViT-based model generalizes better and that the ViT outperforms the CNN on the transparent objects, due to the long-range understanding of the scene.

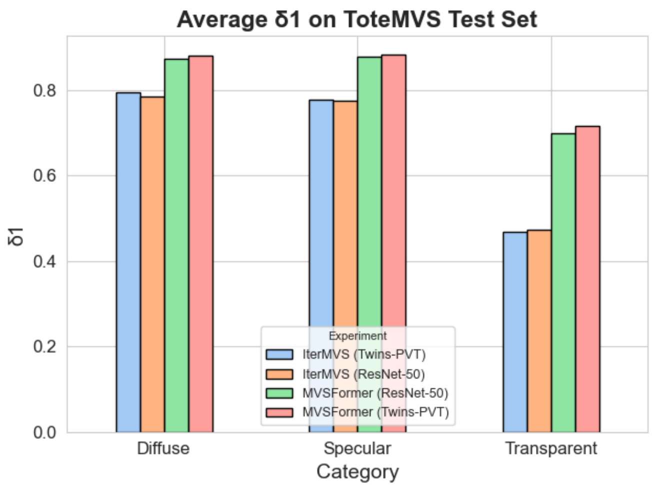


Figure 6: δ_1 for models trained on ToteMVS and evaluated on the test set. MVSFormer performs significantly better on transparent objects.

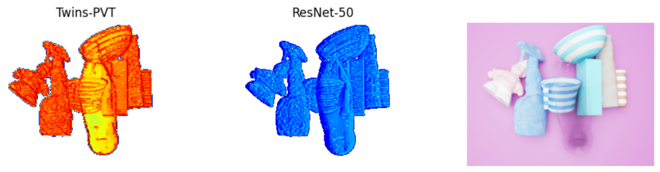


Figure 7: UMAP results on feature maps from both Twins-PVT and ResNet-50. Note how both backbones fail to discern background from foreground on the plastic bottle.

Material	Experiment	Sq.Rel.↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
Diffuse	ResNet-50 (Trained on DTU)	1.5037	0.6847	0.8565	0.8939
	Twins-PVT (Trained on DTU)	0.7437	0.7675	0.8985	0.9278
Specular	ResNet-50 (Trained on DTU)	1.5232	0.6863	0.8552	0.8979
	Twins-PVT (Trained on DTU)	0.7012	0.7453	0.8947	0.9309
Transparent	ResNet-50 (Trained on DTU)	13.5504	0.1164	0.2321	0.3376
	Twins-PVT (Trained on DTU)	12.5421	0.1206	0.2469	0.3595

Table 2: Generalization of models trained on DTU and evaluated on ToteMVS. Note how the metrics on the transparent material are significantly worse compared to the other materials.
$\delta_i = 1.25^i\%$ relative error threshold.

The results of our experiment can be found in Table 2. It becomes immediately apparent how poor the generalization is to transparent objects for *both* backbones, compared to the diffuse and specular objects. Fig. 8 shows how the ViTs predictions are closer to the ground truth, but both struggle greatly with the transparent objects. Interestingly, the ViT is significantly better at generalizing to our diffuse and specular objects than the CNN, which partly confirms what previous literature has to say on the ViTs improved ability to generalize to other domains.

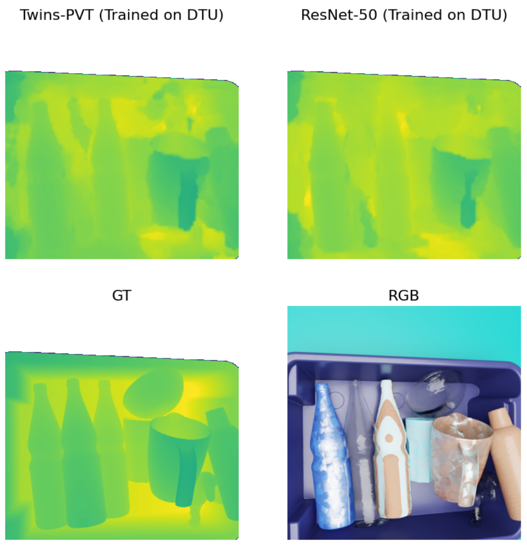


Figure 8: Inference results when MVSFormer is trained on DTU, for the ResNet-50 and Twins-PVT based backbones. Notice how the transparent objects are almost completely ignored.

4.2. Accuracy of ViT and CNN on Specular and Transparent Materials

The texture hypothesis by [12] says that CNNs value object textures more than global object shapes, and according to [30] ViTs perform better than CNNs on shape recognition and dealing with noise and occlusions. While these findings are undoubtedly relevant to object detection tasks, it is unclear whether these properties are relevant for MVS, where the goal is to extract *representative* features of an object surface and not to segment a whole object directly. We hypothesize that specular and transparent objects have confusing (view-dependent) textures and that a network could benefit from recognizing a surface based on shape outline, leading us to think that a ViT-based backbone has the potential to be a superior feature extractor. To answer this question, the ViT- and CNN-based models are trained on ToteMVS until convergence. The goal here is to find out the maximum capability of both configurations to detect the specular and transparent objects.

Our results in Table 6 and Fig. 1 tell a different story than our hypothesis. We observe that the ResNet-50 is nearly just as capable of learning to detect transparent objects. To qualitatively compare the feature maps from both backbones, we apply UMAP [26], a method for reducing the dimensionality of data while maximizing the retention of the original structure. Interestingly, the CNN- *and* ViT-based mapped features in Fig. 7 clearly show that the background behind the transparent object appears through the foreground. Ideally, the feature extractor learns to ignore the background, increasing the ability to match foreground features properly. The results further show that compared to IterMVS, MVSFormer is significantly better at dealing with transparent objects, which cannot be explained by the feature extraction stage as these are identical. This suggests that the 3D CNNs used by MVSFormer are an effective method for transparent objects.

4.3. Robustness to Lack of Textural Features

As mentioned in the related works, for 2D tasks ViTs seem to be more shape-oriented and less sensitive to texture compared to CNNs. In this experiment, the goal is to determine whether these characteristics also hold for MVS feature extraction, where specular and transparent ob-

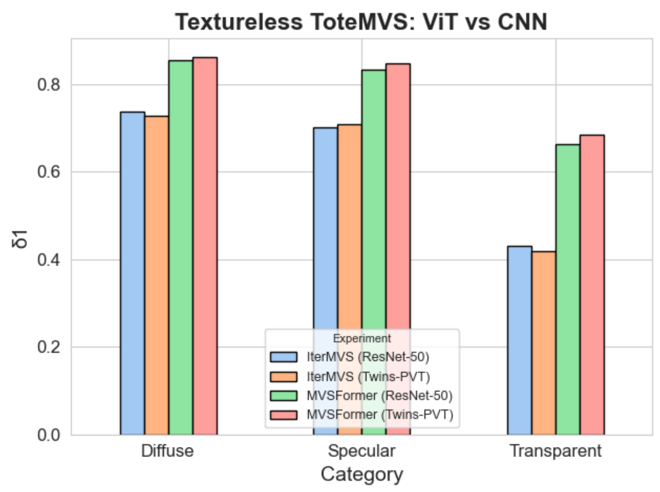


Figure 9: δ_1 score of testing on textureless ToteMVS, after training on the normal (textured) ToteMVS.

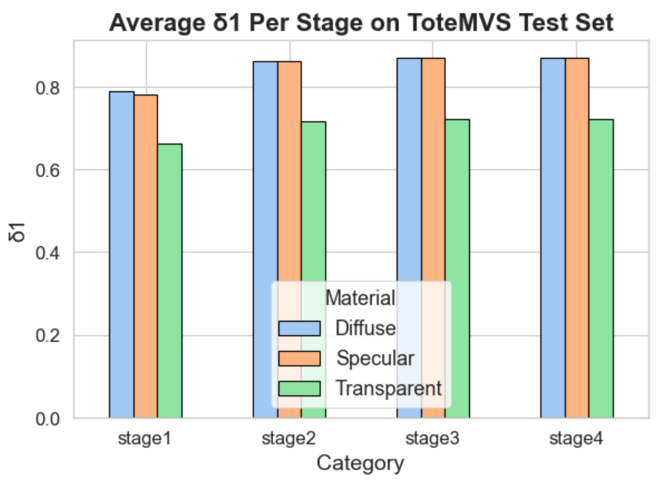


Figure 10: Influence of the individual stages of MVSFormer on the final accuracy per material, trained on ToteMVS and tested on the ToteMVS test set. The stages 1 to 4 refer to the cost volumes depicted in Fig. 3, from the bottom to the top. Note how the last two stages barely influence the final outcome.

jects are considered as having inconsistent textures between views. This leads us to hypothesize that ViTs could extract more representative features on texture-less surfaces. After training on the standard ToteMVS, the model is evaluated against the texture-less ToteMVS (see Fig. 5) to measure any dependency on textural features.

Our results can be seen in Fig. 9 and Table A.4. We find no discrepancy with the original textured results from the previous experiment, indicating that both the ViT and CNN can function when texture and color are removed from the scene.

4.4. Ablations

Problems with Transposed Convolutions. As visually explained by [31], checkerboard artifacts can be the result of transposed convolutions (or 'deconvolutions'). We found that the decoder stage from the ViT part of the backbone suffered from these artifacts, caused by the `ConvTranspose2D` layer from the PyTorch library. Replacing this layer with bilinear upsampling combined with a standard convolutional layer resolves the problem. The feature maps become smoother and the evaluation metrics improve slightly (see Table A.5).

The effect of the refinement stages MVSFormer, like other MVS methods, constructs a coarse-to-fine cost volume by warping features from source views to fronto-parallel planes to the reference camera frame at discrete depth hypothesis intervals. It must be noted that compared to a 2D CNN, 3D CNNs are expensive to run due to the rapid increase of trainable parameters as the volume grows. In Fig. 10 the intermediate δ_1 metrics are visualized per stage and per material. Surprisingly, besides the diminishing effect of the later refinement stages, the final stage only has a detrimental effect. The same effect occurs on the DTU dataset, visualized in Fig. A.1. Every refinement stage in MVSFormer has around 0.3M parameters since the channel size and hypothesis interval are reduced when the resolution increases in order to keep a constant size. Since the final stage is redundant, we run an experiment where we remove the final stage, and 'spend' the 0.3M parameters on the prior stages by increasing the channel sizes to $[80, 40, 20]$ from small to large resolution respectively. We measure a slight overall improvement, as denoted in Table A.5.

The effect of the FPN Combining the ViT features with an FPN is an atypical approach suggested in MVSFormer, which aims to combine a computationally efficient ViT with a standard CNN. We hypothesize that the ViT features can become 'contaminated' with the view-dependent effects that the FPN extracts from the full-scale input images, and that skipping the FPN will be beneficial for depth prediction on transparent materials.

Since the ViT produces features that are at a much smaller scale than the target output, we construct a second decoder network that takes in the ViT features and upsamples them to the appropriate sizes, allowing them to be fed directly into the regularization part. In Fig. 11 we visualize the results. We notice that the error rates decline slightly, together with the accuracy (see Table A.2). We believe that this could be explained by the fact that the network only receives half the original resolution of the input, which causes a coarser matching process.

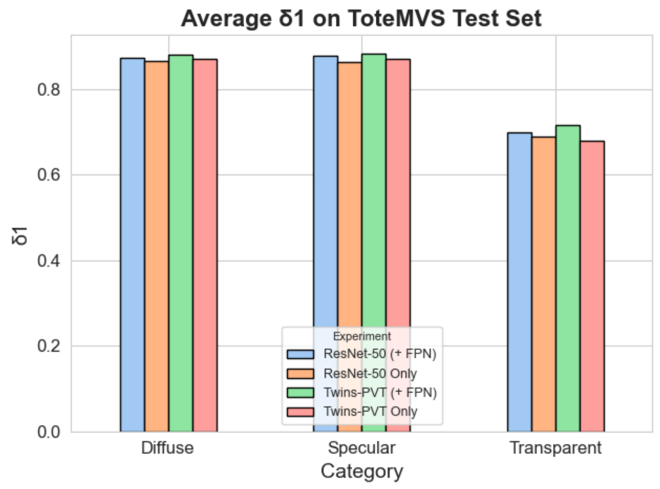


Figure 11: Effect of removing the FPN from both ViT and CNN-based backbones.

5. Discussion

As explained in the related works, ViTs can be more robust and shape-oriented than a similar-sized CNN, and we related this to the domain of MVS. While our results show (both quantitatively and qualitatively) that the Twins-PVT generalizes slightly better for diffuse and specular materials than a ResNet-50, we do not find a significant advantage of the ViT for dealing with specular and transparent objects. The feature maps showed no indication that the ViT dealt with transparent objects differently, as both types showed the background visible through the objects. We want to emphasize that prior works on comparing ViTs to CNNs were mostly on 2D tasks and that our experiments cannot be directly compared to these earlier findings.

Future work We believe it would be interesting to test the differences between 2D tasks (e.g. image segmentation or monocular depth estimation) and the 3D task of MVS. Our hypothesis is that feature extraction for 2D tasks has a different objective, which we believe is producing homogeneous features on similar areas, whereas for MVS the performance would improve when features are as distinctive as possible in order to increase the confidence in matching between views. Perhaps, novel ways can be found that are specialized for this kind of goal.

Additionally, we see a parallel in feature matching across views and self-attention. From a broad perspective, matching a feature (query) to a range of features along the epipolar line (keys) is similar to self-attention. In this work, we solely evaluated the role of a ViT as a feature extractor. Because of the context awareness property, we believe that the principles of transformers could be further applied in different forms to MVS. A method that does something like this is MVSTER [45].

Lastly, the results show a large gap between the performance of MVSFormer and IterMVS on transparent objects, which cannot be explained by the feature extraction backbone. Future work can focus on determining what makes MVSFormer so much better at this task. Since the feature extraction process is identical, the regularization stage is likely responsible for this accuracy gain.

6. Conclusion

In this work, we introduced ToteMVS: a novel, multi-material synthetic MVS dataset that allows for comparison between diffuse, specular and transparent materials in a controlled manner. In this work, we used ToteMVS to find out if a ViT can outperform a similar-sized CNN in the role of a feature extractor of MVS models. We found that transparent objects are significantly more difficult to deal with compared to their diffuse and specular counterparts. While the Twins-PVT (ViT) backbone showed significantly better generalization capability compared to the ResNet-50 (CNN), we found that both achieved similar results when trained on our data. Furthermore, we did not find an advantage of the Twins-PVT when textural features were removed from the scene. Our work is in contrast with the popular belief that ViTs should outperform CNNs, which we hope adds to the ongoing development of future MVS methods.

References

[1] NVIDIA Isaac. Available at `https://docs.nvidia.com/isaac/index.html`, 2023. Accessed on June 13, 2023.

[2] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision*, pages 1–16, 2016.

[3] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, U. Gana, and M. U. Kiru. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, 7:158820–158846, 2019.

[4] C. Cao, X. Ren, and Y. Fu. Mvsformer: Multi-view stereo by learning robust image features and temperature-based depth. *Transactions of Machine Learning Research*, 2023.

[5] J.-R. Chang and Y.-S. Chen. Pyramid Stereo Matching Network, Mar. 2018. arXiv:1803.08669 [cs].

[6] X. Chen, H. Zhang, Z. Yu, A. Opipari, and O. Chadwicke Jenkins. Clearpose: Large-scale transparent object dataset and benchmark. In *European Conference on Computer Vision*, pages 381–396. Springer, 2022.

[7] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021.

[8] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 1). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1, 2022.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[10] Y. Ding, W. Yuan, Q. Zhu, H. Zhang, X. Liu, Y. Wang, and X. Liu. Transmvsnet: Global context-aware multi-view stereo network with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8585–8594, 2022.

[11] Y. Furukawa, C. Hernández, et al. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148, 2015.

[12] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

[13] X. Gu, Z. Fan, S. Zhu, Z. Dai, F. Tan, and P. Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2495–2504, 2020.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[15] M. Ji, J. Gall, H. Zheng, Y. Liu, and L. Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE international conference on computer vision*, pages 2307–2315, 2017.

[16] B. X. D. L. Z. K. Kang Chen, Shaochen Wang and B. Li. Tode-trans: Transparent object depth estimation with transformer. *arXiv preprint arXiv:2209.08455*.

[17] A. Kasper, Z. Xue, and R. Dillmann. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934, 2012.

[18] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, Aug. 2017.

[19] A. Kolesnikov, A. Dosovitskiy, D. Weissenborn, G. Heigold, J. Uszkoreit, L. Beyer, M. Minderer, M. Dehghani, N. Houlsby, S. Gelly, T. Unterthiner, and X. Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[21] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[23] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige. Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11602–11610, 2020.

[24] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[25] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.

[26] L. McInnes, J. Healy, N. Saul, and L. Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[27] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[28] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, Aug. 2020. arXiv:2003.08934 [cs].

[29] Y. Ming, X. Meng, C. Fan, and H. Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021.

[30] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, 34:23296–23308, 2021.

[31] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[32] N. Park and S. Kim. How do vision transformers work? In *International Conference on Learning Representations*, 2022.

[33] S. Paul and P.-Y. Chen. Vision transformers are robust learners. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 36, pages 2071–2081, 2022.

[34] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.

[35] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song. Clear grasp: 3d shape estimation of transparent objects for manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3642. IEEE, 2020.

[36] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[37] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006.

[38] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017.

[39] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[41] F. Wang, S. Galliani, C. Vogel, and M. Pollefeys. IterMVS: Iterative Probability Estimation for Efficient Multi-View Stereo, Dec. 2021. arXiv:2112.05126 [cs].

[42] F. Wang, S. Galliani, C. Vogel, P. Speciale, and M. Pollefeys. PatchmatchNet: Learned Multi-View Patchmatch Stereo. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14189–14198, Nashville, TN, USA, June 2021. IEEE.

[43] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

[44] X. Wang, C. Wang, B. Liu, X. Zhou, L. Zhang, J. Zheng, and X. Bai. Multi-view stereo in the deep learning era: A comprehensive review. *Displays*, 70:102102, 2021.

[45] X. Wang, Z. Zhu, G. Huang, F. Qin, Y. Ye, Y. He, X. Chi, and X. Wang. Mvster: epipolar transformer for efficient multi-view stereo. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI*, pages 573–591. Springer, 2022.

[46] Y. R. Wang, Y. Zhao, H. Xu, S. Eppel, A. Aspuru-Guzik, F. Shkurti, and A. Garg. MVTrans: Multi-View Perception of Transparent Objects, Feb. 2023. arXiv:2302.11683 [cs].

[47] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22–31, 2021.

[48] E. Xie, W. Wang, W. Wang, M. Ding, C. Shen, and P. Luo. Segmenting transparent objects in the wild. *arXiv preprint arXiv:2003.13948*, 2020.

[49] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European conference on computer vision (ECCV)*, pages 767–783, 2018.

[50] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan. Blendedmvs: A large-scale dataset for generalized multi-view stereo networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1790–1799, 2020.

[51] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1592–1599, 2015.

[52] J. Zhang, Y. Yao, S. Li, Z. Luo, and T. Fang. Visibility-aware multi-view stereo network. *arXiv preprint arXiv:2008.07928*, 2020.

[53] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 666–673 vol.1, 1999.

[54] L. Zhu, A. Mousavian, Y. Xiang, H. Mazhar, J. van Eenbergen, S. Debnath, and D. Fox. Rgb-d local implicit function for depth completion of transparent objects, 2021.

[55] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. State of the art on 3d reconstruction with rgb-d cameras. In *Computer graphics forum*, volume 37, pages 625–652. Wiley Online Library, 2018.

A. Additional Tables and Figures

Explanation of metrics used in the following tables:

- RMSE: $\sqrt{\frac{1}{N}\sum_{i\in N}|d_i - d_i^{gt}|^2}$
- RMSE$_{log}$: $\sqrt{\frac{1}{N}\sum_{i\in N}|log(d_i) - log(d_i^{gt})|^2}$
- Abs. Rel: $\frac{1}{N}\sum_{i\in N}\frac{|d_i - d_i^{gt}|}{d_i^{gt}}$
- Sq. Rel: $\frac{1}{N}\sum_{i\in N}\frac{|d_i - d_i^{gt}|^2}{d_i^{gt}}$
- δ-Accuracy: $\%$ of pixels s.t. $max(\frac{d_i}{d_i^{gt}}, \frac{d_i^{gt}}{d_i}) < \delta$

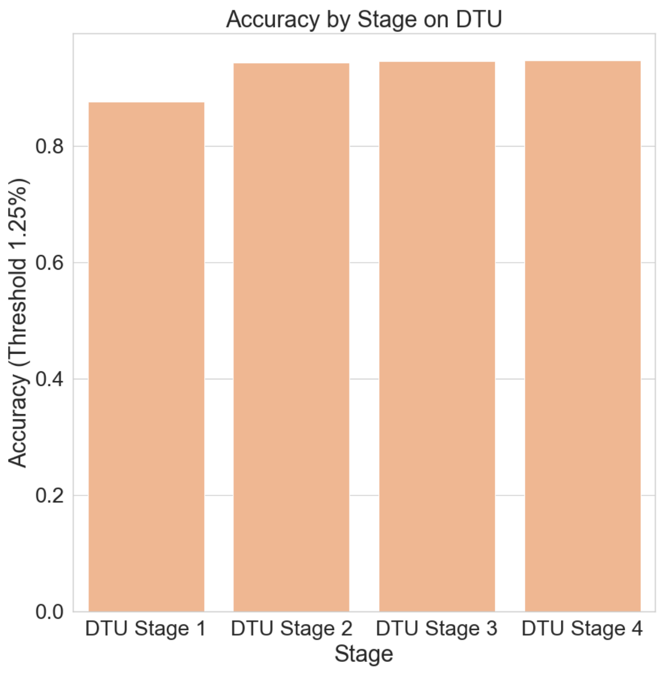


Figure A.1: Influence of stage on the final accuracy per material on DTU [2].

Material	Experiment	A.Rel.↓	Sq.Rel.↓	RMSE↓	RMSE*log*↓	$\delta_1\uparrow$	$\delta_2\uparrow$	$\delta_3\uparrow$
Diffuse	IterMVS (ResNet-50)	0.1366	0.8923	22.2930	0.2628	0.7864	0.8855	0.9181
	IterMVS (Twins-PVT)	0.1340	0.8785	23.0048	0.2688	0.7944	0.8883	0.9183
	MVSFormer (ResNet-50)	0.0983	0.5382	18.5555	0.2137	0.8730	0.9224	0.9413
	MVSFormer (Twins-PVT)	0.0953	0.5568	19.4142	0.2230	0.8804	0.9206	0.9399
Specular	IterMVS (ResNet-50)	0.1386	0.8977	21.3466	0.2522	0.7758	0.8831	0.9164
	IterMVS (Twins-PVT)	0.1488	1.0386	24.0194	0.2839	0.7770	0.8776	0.9097
	MVSFormer (ResNet-50)	0.0943	0.4920	17.8147	0.2073	0.8783	0.9282	0.9482
	MVSFormer (Twins-PVT)	0.0937	0.5455	19.0367	0.2185	0.8825	0.9257	0.9446
Transparent	IterMVS (ResNet-50)	0.2750	1.9467	34.4508	0.4069	0.4731	0.6899	0.7930
	IterMVS (Twins-PVT)	0.2964	2.2888	37.5437	0.4445	0.4689	0.6763	0.7770
	MVSFormer (ResNet-50)	0.1623	0.9627	24.8356	0.2868	0.6992	0.8525	0.9007
	MVSFormer (Twins-PVT)	0.1518	0.8555	24.1044	0.2785	0.7159	0.8620	0.9098

Table A.1: Results on ToteMVS for the different materials. Pretrained backbones are used.
$\delta_i = 1.25^i\%$ relative error threshold.

Material	Experiment	A.Rel.↓	Sq.Rel.↓	RMSE↓	RMSE*log*↓	$\delta_1\uparrow$	$\delta_2\uparrow$	$\delta_3\uparrow$
Diffuse	ResNet-50 (no FPN)	0.0996	0.5456	18.4374	0.2134	0.8674	0.9207	0.9410
	ResNet-50 (with FPN)	0.0983	0.5382	18.5555	0.2137	0.8730	0.9224	0.9413
	Twins-PVT (no FPN)	0.0948	0.4900	18.2373	0.2103	0.8722	0.9227	0.9437
	Twins-PVT (with FPN)	0.0953	0.5568	19.4142	0.2230	0.8804	0.9206	0.9399
Specular	ResNet-50 (no FPN)	0.0961	0.4790	17.6827	0.2044	0.8632	0.9289	0.9498
	ResNet-50 (with FPN)	0.0943	0.4920	17.8147	0.2073	0.8783	0.9282	0.9482
	Twins-PVT (no FPN)	0.0945	0.4823	17.9918	0.2076	0.8712	0.9273	0.9477
	Twins-PVT (with FPN)	0.0937	0.5455	19.0367	0.2185	0.8825	0.9257	0.9446
Transparent	ResNet-50 (no FPN)	0.1584	0.8742	23.8879	0.2776	0.6893	0.8544	0.9067
	ResNet-50 (with FPN)	0.1623	0.9627	24.8356	0.2868	0.6992	0.8525	0.9007
	Twins-PVT (no FPN)	0.1567	0.8036	23.4757	0.2726	0.6804	0.8507	0.9055
	Twins-PVT (with FPN)	0.1518	0.8555	24.1044	0.2785	0.7159	0.8620	0.9098

Table A.2: Measuring the effect of the FPN on MVSFormer for both ViT and CNN.
$\delta_i = 1.25^i\%$ relative error threshold.

Experiment	A.Rel.↓	Sq.Rel.↓	RMSE↓	RMSE*log*↓	$\delta_1\uparrow$	$\delta_2\uparrow$	$\delta_3\uparrow$
Twins-PVT (Original Weights)	19.8490	0.0071	1.5927	0.0255	0.9461	0.9607	0.9677
Twins-PVT (Reproduced)	19.6378	0.0076	1.5583	0.0253	0.9440	0.9589	0.9656
ResNet-50 (Reproduced)	18.6527	0.0072	1.0488	0.0242	0.9327	0.9528	0.9614

Table A.3: Reproduction experiment of MVSFormer on DTU. Only Twins-PVT weights were shared by the original author.
$\delta_i = 1.25^i\%$ relative error threshold.

Material	Experiment	A.Rel.↓	Sq.Rel.↓	RMSE↓	RMSElog↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
Diffuse	IterMVS (ResNet-50)	0.1557	1.2405	23.1547	0.2755	0.7376	0.8686	0.9124
	IterMVS (Twins-PVT)	0.1794	1.3888	25.8630	0.3114	0.7272	0.8544	0.8919
	MVSFormer (ResNet-50)	0.1032	0.5467	19.3285	0.2227	0.8530	0.9129	0.9377
	MVSFormer (Twins-PVT)	0.0999	0.5412	19.3313	0.2212	0.8604	0.9167	0.9400
Specular	IterMVS (ResNet-50)	0.1673	1.1327	23.8771	0.2834	0.7006	0.8540	0.9024
	IterMVS (Twins-PVT)	0.1678	1.1664	26.1316	0.3056	0.7070	0.8494	0.8947
	MVSFormer (ResNet-50)	0.1157	0.6903	20.4088	0.2370	0.8324	0.9057	0.9311
	MVSFormer (Twins-PVT)	0.1111	0.6992	20.3546	0.2350	0.8479	0.9103	0.9334
Transparent	IterMVS (ResNet-50)	0.3106	2.6305	37.2582	0.4453	0.4297	0.6514	0.7650
	IterMVS (Twins-PVT)	0.3370	3.2698	41.7973	0.4924	0.4177	0.6356	0.7414
	MVSFormer (ResNet-50)	0.1837	1.2503	27.3410	0.3160	0.6620	0.8260	0.8819
	MVSFormer (Twins-PVT)	0.1705	1.0626	25.8622	0.2985	0.6834	0.8416	0.8945

Table A.4: Results on textureless ToteMVS for the different materials. Pretrained backbones are used. $\delta_i = 1.25^i\%$ relative error threshold.

Material	Experiment	Sq.Rel.↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
Diffuse	ResNet-50	0.5382	0.8730	0.9224	0.9413
	Twins-PVT (3 Stages)	0.5435	0.8838	0.9223	0.9408
	Twins-PVT (Bilinear + Conv)	0.4821	0.8873	0.9273	0.9456
	Twins-PVT (Default)	0.5568	0.8804	0.9206	0.9399
Specular	ResNet-50	0.4920	0.8783	0.9282	0.9482
	Twins-PVT (3 Stages)	0.5135	0.8837	0.9267	0.9453
	Twins-PVT (Bilinear + Conv)	0.4749	0.8864	0.9310	0.9496
	Twins-PVT (Default)	0.5455	0.8825	0.9257	0.9446
Transparent	ResNet-50	0.9627	0.6992	0.8525	0.9007
	Twins-PVT (3 Stages)	0.7771	0.7225	0.8685	0.9149
	Twins-PVT (Bilinear + Conv)	0.7307	0.7228	0.8688	0.9158
	Twins-PVT (Default)	0.8555	0.7159	0.8620	0.9098

Table A.5: Overview of ablation effects on MVSFormer, trained and tested on ToteMVS. $\delta_i = 1.25^i\%$ relative error threshold.

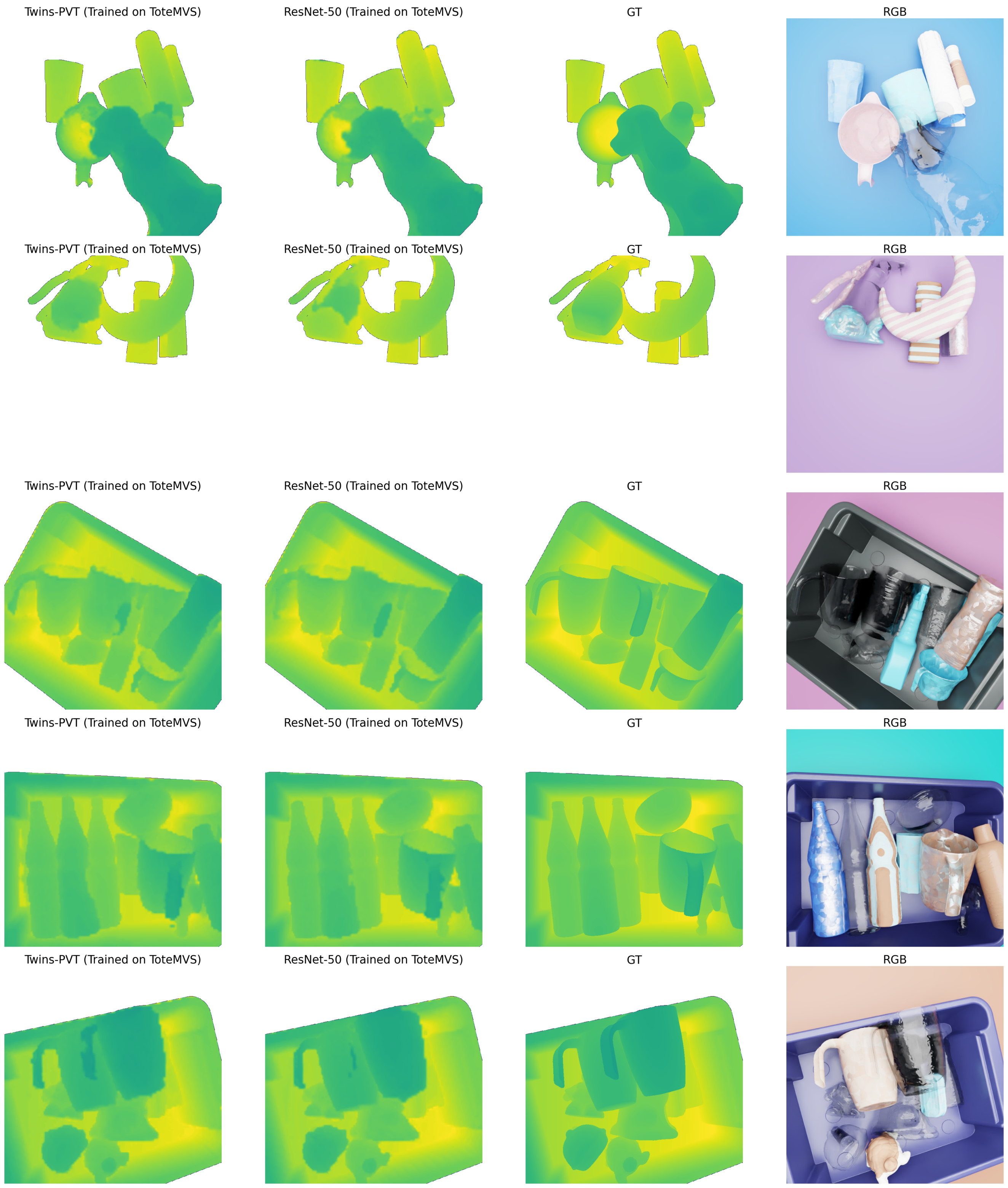

Twins-PVT (Trained on ToteMVS)
ResNet-50 (Trained on ToteMVS)
GT
RGB
Twins-PVT (Trained on ToteMVS)
ResNet-50 (Trained on ToteMVS)
GT
RGB
Twins-PVT (Trained on ToteMVS)
ResNet-50 (Trained on ToteMVS)
GT
RGB
Twins-PVT (Trained on ToteMVS)
ResNet-50 (Trained on ToteMVS)
GT
RGB
Twins-PVT (Trained on ToteMVS)
ResNet-50 (Trained on ToteMVS)
GT
RGB

Figure A.2: Twins-PVT vs ResNet-50 based depth predictions using MVSFormer[4] trained on ToteMVS.

3

Supplementary Material

3.1. Mathematical Foundations

In order to understand how multiple views are combined in order to predict depth, it is important to understand the underlying mathematics first, which comes down to basic linear algebra. In this section, we briefly summarize some of the linear algebra that is used in further sections.

3.1.1. Basic Vector Operations

- **Dot product**: the dot product between two vectors a and b of length n is given by $a \cdot b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$. The dot product is only defined for vectors of equal length.
- **Matrix multiplication**: to multiply two matrices A of size (m, n) and B of size (n, k), you construct a new matrix of size (m, k) such that every element is the dot product between a column of A and a row of B. Mathematically, this can be written as $C = A \times B$ such that $C_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}$.
- **Inverse matrices**: the inverse of a matrix A, denoted as A^{-1}, is a matrix such that $AA^{-1} = I$ where I is the identity matrix.

3.1.2. Basis and Coordinate Systems

A collection of points can be described using a coordinate system, most often in 2D or 3D space. All points have a relative position to each other and can be viewed from different positions or angles, just like in the real world. The default point of reference, or **standard basis**, is defined by the identity matrix forming the basis vectors. Any vector or point in this space can be described as a linear combination of these basis vectors. Defining this is useful when we need to change our coordinate system, which could be described by moving a camera around the scene. Doing so changes the position of the points in space relative to the camera, but not relative to each other.

Switching between frames of reference is easy, as all we need is the basis of the two reference frames. For example, take references 1 and 2 described by E_1 and E_2. To go from reference system 1 to 2, we first move to the standard basis by computing the inverse of E_1, and then move to the basis of 2 by multiplying with E_2. Combined, we simply multiply every point in our space with $E_2 E_1^{-1}$.

3.1.3. Rigid Transformations

For moving a camera around a scene, two primitive operations are important: rotating and translating (moving). These kinds of transformations are called **rigid transformations** since the relative position between any point in the vector space remains the same. Here, we assume a 3D space. Given a rotation defined by 3x3 matrix R, and a translation defined by 1x3 vector T, any point can be transformed using this equation: $u_2 = Ru_1 + T$. To simplify this equation, we can write it down using the **augmented form**,

$$A = \left[\begin{array}{c|c} R & T \end{array}\right] = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

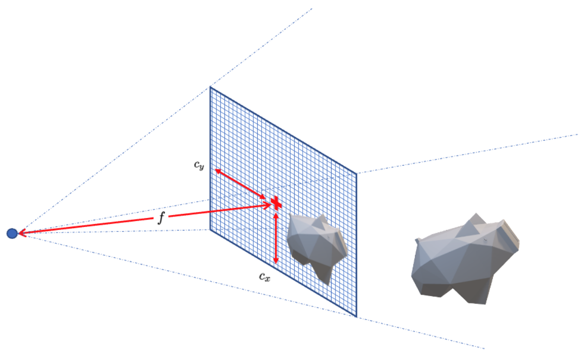


Figure 3.1: An example camera frustum describing the pinhole model.

To use the augmented matrix, any point that is being multiplied by it must also be augmented, meaning $u_a = [u|1]$. It is trivial to see that $Au_a = Ru + T$. A useful property of the augmented form is that it is easier to invert transformations compared to the prior form. We can now describe a camera's position and viewing direction using the augmented form.

3.2. Stereo Depth Estimation Fundamentals

3.2.1. The Pinhole Camera Model

The pinhole camera model (see Fig. 3.1) is a simplified model of how a camera sensor perceives the 3D world on a 2D surface (the image plane), by modeling how light travels through a small hole (pinhole). Even though it is a relatively simple model, it provides a useful and intuitive way of dealing with world-to-camera (w2c) and camera-to-world (c2w) projections. This model uses the following matrices:

Intrinsic matrix

$$K = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Where:

f_x : Focal in x direction in pixels / mm
f_y : Focal in y direction in pixels / mm
s : Axis skew (often 0)
c_x : Center offset in x
c_y : Center offset in y

Extrinsic matrix

$$\left[\begin{array}{c|c} R & T \end{array}\right]$$

Where:

R : Rotation matrix
T : Translation vector
Note : Matrix in augmented form

Using the above matrices, we can calculate the **camera matrix** M, which converts 3D world coordinates to 2D pixel coordinates on the camera image plane.

$$M = K \left[\begin{array}{c|c} R & T \end{array}\right]$$

Conversely, the intrinsic and extrinsic matrices can be used to map 2D pixel coordinates to the 3D world. Note that because of the augmented form of the camera matrix, coordinates must also be in augmented form.

3.2.2. Mapping Pixels to Pixels Using Depth

In the scenario where we have multiple images of the same scene accompanied by the exact camera extrinsic and intrinsic matrices, we can warp one viewpoint to the other using the pinhole camera model

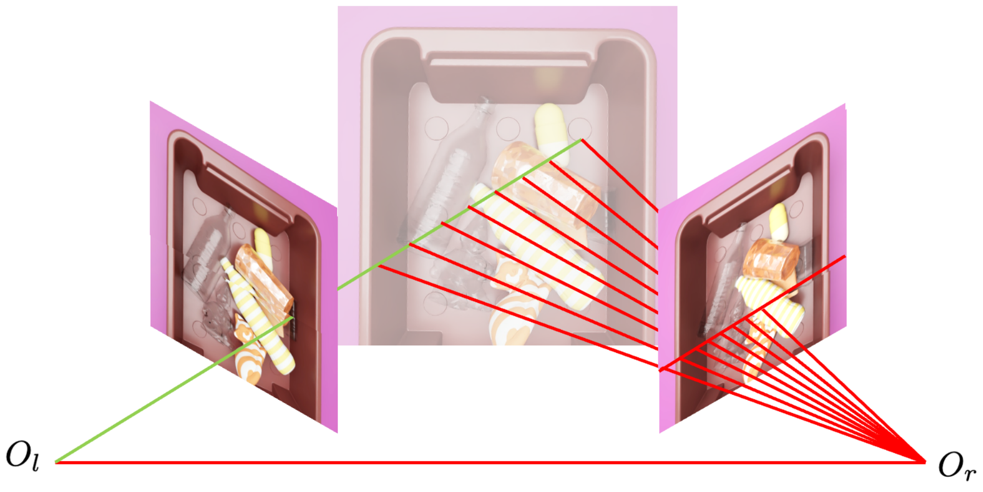


Figure 3.2: The plane defined by the two optical centers and the query vector define the epipolar search line.

described in the previous section. Note that for this to work, we need an accurate depth map for I_1, denoted as d in the following transformation.

Given two images with an estimated depth per pixel z_i, intrinsic matrices K_1, K_2 and extrinsic matrices E_1, E_2, we can map a pixel coordinate from I_1, written as u_1 in augmented form to the coordinate system of I_2, written as u_2 with the following transformation:

$$W \approx E_2 E_1^{-1} \begin{bmatrix} (K_1^{-1} u_1) z_i \\ 1 \end{bmatrix}$$

$$u_2^{'} \approx K_2 \begin{bmatrix} W_x / W_z \\ W_y / W_z \\ 1 \end{bmatrix}$$

This series of transformations can be broken down into three steps:

1. $(K_1^{-1} u_1) z_i$: Project the 2D points to the image plane in 3D from the reference point of camera 1. Multiply by the depth values per pixel to get the 3D positions from the camera's 1 point of view.
2. $E_2 E_1^{-1}$: Change the basis of the 3D coordinates by computing the relative difference between cameras 1 and 2.
3. $K_2 \begin{bmatrix} W_x / W_z \\ W_y / W_z \\ 1 \end{bmatrix}$: First project the points onto the image plane by dividing by their z values. Finally, obtain their 2D pixel coordinates by multiplying them with the intrinsic matrix of camera 2.

Since z_i is an estimation, the precision of the mapping depends on the precision of this estimation. Given two depth predictions, the consistency can be evaluated by warping between pairs of images' points of view and calculating the error.

The principles above can be used inversely: given an estimated mapping from $u_1 \rightarrow u_2^{'}$, we can calculate a depth estimation z_i. This is the fundamental mechanic behind MVS depth estimation.

3.2.3. Classic Epipolar Geometry

In the previous part, we explained how you can perfectly map viewpoints from one to the other if you have access to an accurate depth prediction, and how an inaccurate depth map will lead to misaligned points in the mapping. Aligning points such that they do match is the core principle behind epipolar matching

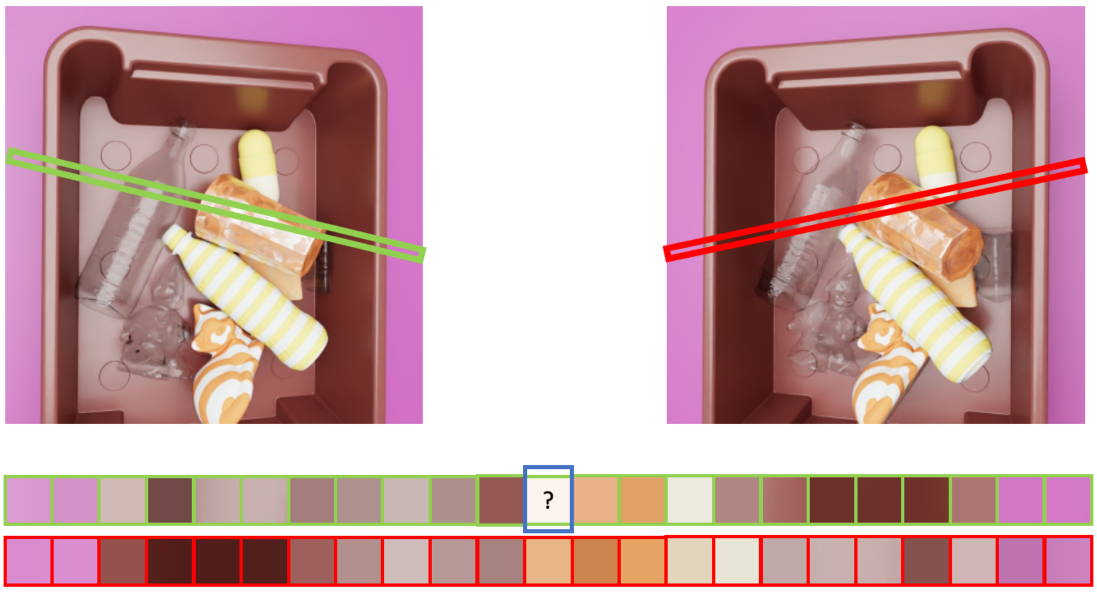


Figure 3.3: The 1D search space for any epipolar matching algorithm on the pixel level (simplified representation).

strategies, that predict depth values based on correspondence between features or pixels. The amount of which a surface point moves between views is also often called **disparity**. A naive way of predicting disparity would result in a 2D search space (across the whole image). Here, we explain how epipolar geometry can drastically decrease this search space.

Given two cameras with optical centers O_l, O_r, we can draw a baseline between the two points (see Fig. 3.2). Next, we select a query point in one of the image planes, through which we shoot a ray from the optical center through the designated pixel. As the figure shows, the baseline and the query ray form a plane on which the **epipolar line** is defined as the intersection of the epipolar plane with the reference image plane.

The epipolar line restricts the search of a corresponding surface point from a 2D space to a (much smaller) 1D space (see Fig. 3.3). Even though this seems to solve all of our problems, the next section will explain why this method is far from perfect.

3.2.4. Challenges with Epipolar Matching

This part highlights some of the main problems with epipolar matching, describing why estimating depth with epipolar matching can lead to ambiguous situations.

Occlusions appear when one camera can see something that the other cannot. Depending on the scene, increasing the baseline impacts the chance of occlusions between views.

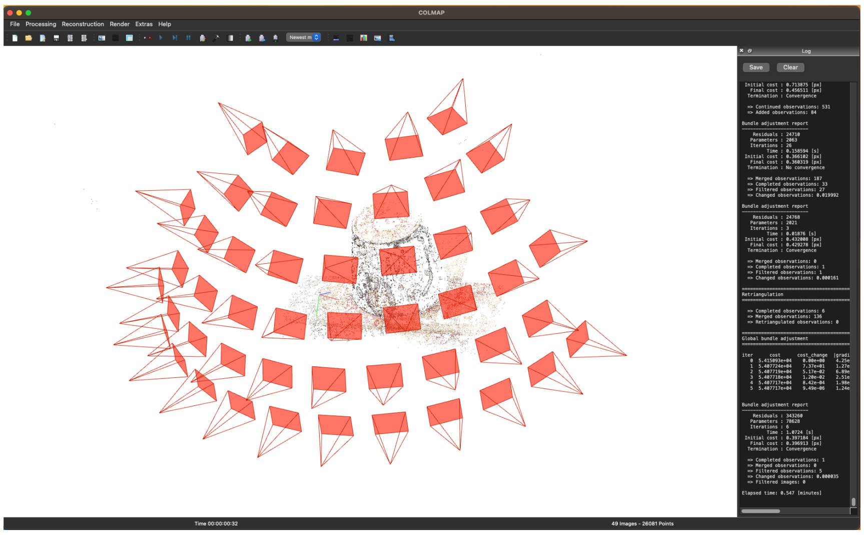
Textureless regions cause ambiguity when matching pixels along the epipolar line, due to the fact that just based on pixel color multiple options can seem optimal.

View-dependent effects are like the opposite problem of the prior case, because now the surface changes depending on the light and viewing direction, making it much harder to match pixels. This effect is especially strong on transparent and highly reflective surfaces.

3.2.5. Camera setup

Even though there are an infinite number of ways one could set up a stereo pair of cameras, there are trade-offs to be made.

1. **Horizontally aligned parallel cameras** lead to perfectly horizontal epipolar lines. This has the advantage of reduced computational complexity since one would only need to find correspondence along the horizontal axis of the image. The downside is that both cameras see parts of the scene that the other cannot, depending on the baseline.



```
 Initial cost : 0.713875 [px]
   Final cost : 0.456511 [px]
  Termination : Convergence

  => Continued observations: 531
  => Added observations: 84

Bundle adjustment report
------------------------
    Residuals : 24710
   Parameters : 2063
   Iterations : 26
         Time : 0.158594 [s]
 Initial cost : 0.366102 [px]
   Final cost : 0.360319 [px]
  Termination : No convergence

  => Merged observations: 187
  => Completed observations: 33
  => Filtered observations: 27
  => Changed observations: 0.019992

Bundle adjustment report
------------------------
    Residuals : 24768
   Parameters : 2021
   Iterations : 3
         Time : 0.01876 [s]
 Initial cost : 0.432008 [px]
   Final cost : 0.429278 [px]
  Termination : Convergence

  => Merged observations: 0
  => Completed observations: 1
  => Filtered observations: 1
  => Changed observations: 0.000161

========================================
Retriangulation
========================================

  => Completed observations: 6
  => Merged observations: 136
  => Retriangulated observations: 0

========================================
Global bundle adjustment
========================================

iter      cost      cost_change  |gradi
   0  5.415093e+04    0.00e+00    4.25e
   1  5.407724e+04    7.37e+01    1.27e
   2  5.407719e+04    5.17e-02    6.89e
   3  5.407718e+04    1.20e-02    2.51e
   4  5.407717e+04    8.42e-04    1.98e
   5  5.407717e+04    9.49e-06    1.24e


Bundle adjustment report
------------------------
    Residuals : 343260
   Parameters : 78628
   Iterations : 6
         Time : 1.0724 [s]
 Initial cost : 0.397184 [px]
   Final cost : 0.396913 [px]
  Termination : Convergence

  => Completed observations: 1
  => Merged observations: 0
  => Filtered observations: 5
  => Changed observations: 0.000035
  => Filtered images: 0

Elapsed time: 0.547 [minutes]
```

Figure 3.4: COLMAP scene reconstruction example on DTU scene 1.

2. **A strong toe-in** will ensure both cameras see the same object, which is important for the matching part. Note that a strong toe-in will increase the amount of occlusion depending on how close an object is to the camera and the baseline. A strong toe-in also results in a keystone distortion, meaning the warped views will be more like a trapezoid than a rectangular plane.

Depending on the situation and requirements (number of cameras, distance to object, completeness to name a few) one should decide on the optimal setup for the cameras.

3.3. Structure from Motion

Structure from Motion (SfM) is a method for estimating 3D structures from multiple 2D images, while simultaneously estimating camera positions and orientations. Whereas modern deep-learning-based MVS methods focus on estimating depth based on disparity and known camera poses, giving a more complete view, SfM is typically less complete but extremely accurate. This makes it a useful tool for estimating camera orientations that can be used in other MVS methods.

The SfM problem can be formulated as: given a pair of images, how can we extract the relative camera position and orientations and reconstruct the 3D geometry of the scene? The first work to address this problem was by [7], who introduced a method for estimating stereo camera orientation based on matching points between pairs, later called the *eight-point algorithm*. This method is known to be sensitive to noise but was a first step in the right direction.

SfM assumes that the camera intrinsics are known, such as the focal length, skew, and camera offset. These intrinsics can be estimated using the Zhang method [8], which can estimate the parameters based on images from a known checkerboard pattern.

Extracting features is a critical step in SfM. Features are optimal when they can be extracted consistently between views, and do not change depending on the viewing direction (for example with reflective surfaces). Texture variation is also important, as extracting features on textureless surfaces is difficult.

Using the learned features, 3D structure can be estimated through triangulation. Once initial 3D points and camera positions are estimated, they are optimized to minimize the reprojection error. This produces a sparse, but accurate point cloud as shown in Fig. 3.4. A practical environment for running state-of-the-art SfM algorithms is COLMAP. COLMAP [9] is an open-source all-in-one SfM and MVS utility that is widely used by researchers and developers. It is important to note that COLMAP deploys a wide range of techniques that others created.

Fig. 3.4 shows the end-product of a scene reconstruction using the GUI version of COLMAP. Through the CLI version, this process can be automated as a preprocessing step, for example, to extract camera positions and orientations.

3.4. Deep Learning

In this section, we go over the core principles behind deep learning with a focus on computer vision. After reading this chapter, you should hopefully have a clear idea of why deep learning should be able to overcome the challenges with classical stereo-vision matching methods.

3.4.1. Core Principles

Deep learning methods allow us to learn patterns in our data without explicitly telling it how to do so. In the most simple terms, we feed a neural network input data and compare the predicted output to our target output. The word ”deep” comes from the number of layers that are used in the network. To understand why deep neural networks exist, we first start at the beginning.

The Perceptron

The perceptron [10] was originally inspired by the neurons of the brain, which learn by regularly activating specific neural pathways. A perceptron can have 1 or more inputs and can be used as a trainable binary classification model. The formal definition is as follows:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

The perceptron forms a plane in the space of possible inputs, where anything above the plane would be classified as 1 and anything below the plane as 0. While the perceptron was one of the early steps to modern artificial intelligence, it had big shortcomings. Most notably, the perceptron is not able to learn the XOR function due to its inherent linear shape.

Loss Functions

In order for any model to ”learn”, its weights must be adjusted or nudged in the right direction such that the difference between the model output and our target output becomes smaller. To adjust the weights, we must figure out in what direction to move them. The right direction can be found by taking the derivative of the loss function with respect to the model parameters. This technique is called gradient descent. Let us first give the formal definition of a loss function:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i; \theta))$$

The loss function gives the average loss per (x_i, y_i) pair taken from our training data. A popular loss function is the Mean-Squared Error (MSE):

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

In order to determine the right direction for our model parameters to move in, we find the derivative of the MSE in the case of a simple perceptron with respect to the weights. Note that we simplify the notation $\sum_{i=1}^{n} w_i x_i + b = W^T X + b$.

$$\frac{\delta y}{\delta W} = \frac{2}{N} \sum_{i=1}^{N} (y_i - W^T X_i - b)(-X_i)$$

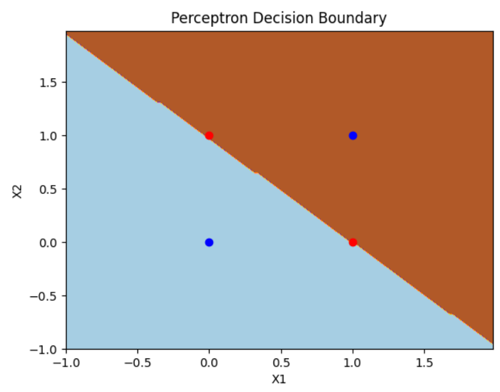
Using this formula we can now calculate the gradient of the loss function with respect to the weights. Since our goal is to minimize the loss, we correct the weights by a factor of the opposite direction. This factor is also called the learning rate ϵ:

$$\theta' = \theta - \epsilon \frac{\delta y}{\delta \theta}(x; y; \theta)$$

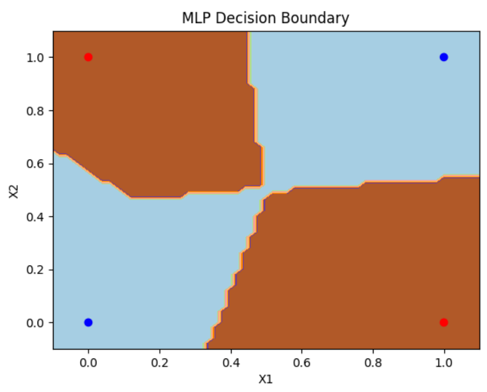
This process can be repeated until the model converges or after a fixed number of iterations. The learning rate is an important hyperparameter that heavily influences whether the model can find it’s global minimum. If the learning rate is too high, the model may skip over the global minimum and fail to reach it.

If the learning rate is too low, the model may get stuck in a local minimum. There is no clear rule for the optimum learning rate, but there are strategies that can give a performance boost (read more about it in the subsection about optimizers).

The Multi-Layer Perceptron



(a) Classic perceptron is unable to perfectly divide the two classes



(b) MLP with ReLU and Sigmoid activation function

Figure 3.5: Learning the XOR function

The Multi-Layer Perceptron (MLP) was proposed by the same author years later and solves the issue of linearity by adding hidden layers with nonlinear activation functions in between. The hidden layer can be described using the following formula:

$$f_i^l(x) = \sigma(w_0^l + \sum_{n=1}^{N} w_n^l f_n^{l-1})$$

The activation functions allow the decision boundary to become nonlinear (see Fig. 3.5), giving the model more flexibility. See Fig. 3.6 for an overview of the most commonly used activation functions.

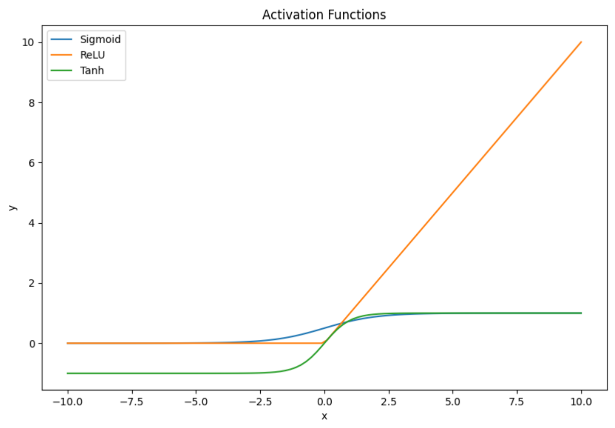


Figure 3.6: Common activation functions used in deep learning.

Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an optimization of the standard gradient descent, where the objective is to find the optimal weights as quickly as possible. Whereas classic gradient descent calculates the gradient based on the whole training data, SGD splits the data into ”batches” with a fixed size. On every iteration, a batch is randomly selected, and only the resulting loss is used to update the weights. This way, weights are updated more often, and more noise is added allowing weights to escape local minima. There are many other optimization methods, such as RMSProp or Adam, which are more advanced methods that we suggest the reader look into when designing a training method.

3.4.2. Convolutional Neural Networks

Detecting patterns in images through the use of classic MLP networks has limitations, but has notoriously been demonstrated to work for simple black-and-white images to detect single-handwritten digits. Simply by taking every row of pixels and concatenating them, a network can be built that accepts every pixel as input. This is far from ideal: the network would overfit the training data for more complex cases, and since colored images are usually composed of red, green, and blue (RGB) channels, the number of learnable weights explodes as well. The convolutional neural network [11] (CNN) was able to overcome these problems and has become a fundamental element in modern computer vision.

Instead of having a weight for every input, square $n \times n$ kernels are convoluted over the input image, thereby transforming the raw image into features. As a very basic example, take the following kernel:

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

The star $\star$ notation is typically used to indicate that a kernel is applied to a matrix. The operation translates to the following mathematical expression:

$$(I \star K)_{ij} = \sum_{a=1}^{n} \sum_{b=1}^{n} I_{i+a-1,j+b-1} \cdot k_{ab}$$

Note that applying a $n \times n$ kernel to a matrix will shrink the width and height by $(n-1)$. To maintain the same size, the image can be *padded* with values according to some strategy (e.g. constant values). Applying the kernel from the example to the left image in Fig. 3.7 produces the image to the right. This is a good example of feature extraction: in this case, the kernel extracts the right edges in the picture. Usually, CNNs start with detecting edges in the first layer, and as the network gets deeper these features are combined to detect shapes and finally things.

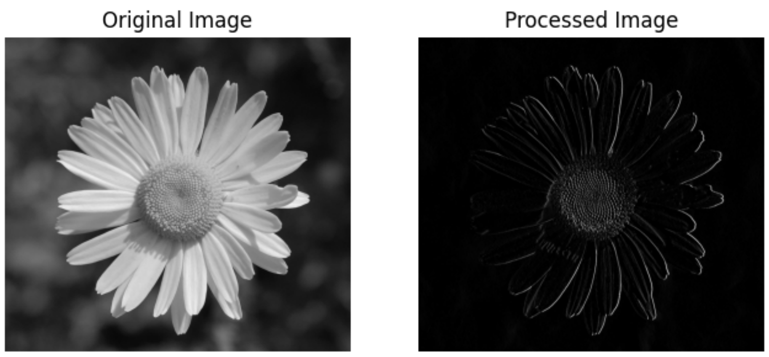


Figure 3.7: Result after applying the edge detection kernel to a sample image.

Imagine that not just one, but many kernels are learned and applied to the input. Due to the large amount of data this produces, we need some way to condense this information in a more compact format. One of the methods of reducing the dimensions of the features is called *pooling* and is another essential step in making CNNs work. An often used pooling operation is called *MaxPooling* which simply takes the maximum value from every cell in a discrete grid of the feature channel. This grid is determined by the size of the pooling operation: a 2×2 operation downsizes the feature map by a factor of 2. Another technique that allows the shrinking of input is called *stride*, which indicates by how many pixels the convolution window moves every step. A stride of 2 would result in a feature map that is roughly twice as small in both dimensions.

By chaining convolutions, we start capturing increasingly more high-level features because of the increased *receptive field* of a single pixel in the feature map. The reason for this is that the output of a $n \times n$ convolution ”sees” a range of $n \times n$ pixels per output pixel. Compound this and you can see how the deeper you go, the wider the visibility or receptive field of every feature element.

Note that kernels are translation invariant: meaning it does not matter where in the image the feature is located, the kernel will extract it regardless. This is an important property that reduces the chances of the network overfitting on exact object locations. A typical CNN is not *scale invariant* however; this can result in overfitting on objects of specific scales.

3.4.3. Learned Upsampling for 2D Data

Reducing the resolution of an image with learned convolutions allows the network to distill information about a region of the data in a compact manner. For many applications, the desired output has the same resolution as the input, for example in semantic segmentation or depth estimation. Traditional upsampling methods, like nearest-neighbor, bilinear, or bicubic interpolation are ’dumb’ in the sense that they do not allow the network to learn an interpretation of what the upsampled data should look like. Just like we can use convolutions to learn appropriate downsampling, we can also do the inverse, often called transposed convolutions or deconvolution. In transposed convolutions, we still have a kernel, but instead of taking a region of the input and multiplying it with the kernel, we pick a single input value and spread it over a larger spatial region in the output. Depending on the kernel size, the input is padded around every element, and the kernel is multiplied with every single value and added to the resulting feature map. Note that this operation can create ’checkerboard’ artifacts, which can look like the example in Fig. 3.8. These can be partly caused by uneven overlaps of the convolutions, which occur when the kernel size is not divisible by the stride. Another way to get rid of checkerboard artifacts is to avoid using a transposed convolution, and instead, first upsample the image using bilinear interpolation and then apply a standard convolution.

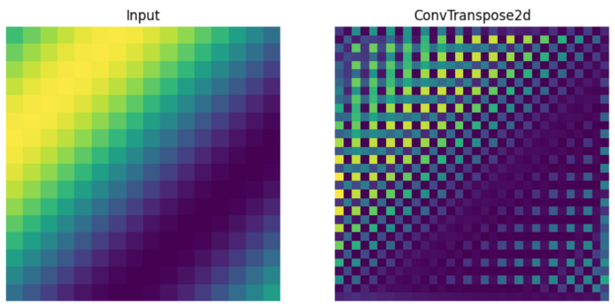


Figure 3.8: Checkerboard artifact from the deconvolution operation.

3.4.4. Feature Pyramid Networks

Dense prediction tasks are tasks that require a prediction of some value (discrete or continuous) per pixel. Often, it makes sense to reduce the spatial dimensionality and increase the channel size of the latent space, in order to learn many representations of an image. In order to project this high-level understanding of a scene onto a format that will allow us to perform per-pixel prediction, we can use *Feature Pyramid Networks* (FPNs) [12]. Using this technique, we can essentially enrich the features at a low-level with high-level features. An example architecture is depicted in Fig. 3.9. This network outputs four levels, which many downstream tasks use to go from a coarse to fine estimation.

3.4.5. Gated Recurrent Unit Networks

Gated Recurrent Unit (GRU) Networks are a subclass of recurrent neural networks, meaning they can process sequential information and pass intermediate output along through a hidden state. An example of sequential information is a time series (e.g. the price of a stock), but it can also be used to process sequential camera frames, thereby modeling spatial and temporal relationships between objects. Classic

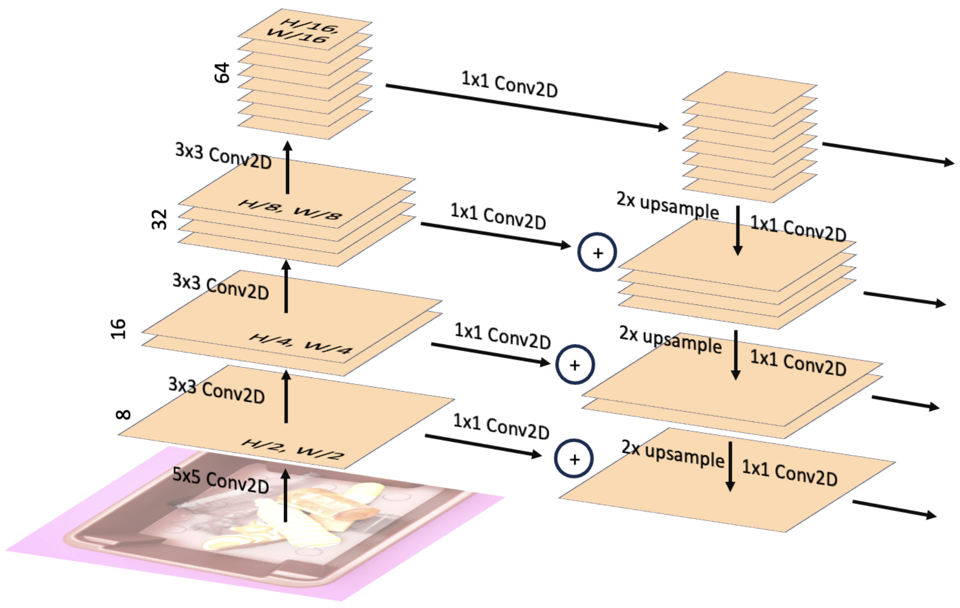


Figure 3.9: General architecture of a Feature Pyramid Network with 4 levels.

recurrent networks suffer from the vanishing gradient problem: when gradients of the loss function get close to zero with respect to the weights of the early elements in the recurrent chain. This has the negative effect that the weights of early layers or blocks are updated much slower than the final layers.

To mitigate this problem, GRUs implement changes to allow the network to control the amount of information that is passed along to the next layers. Through mathematical constructs called ”gates”, the network can decide how much of the previous hidden state to pass along (update) or how much to forget (reset). Formally, a complete GRU is described with the following formulas:

Formulas:

$$\begin{aligned} z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\ \hat{h_t} &= \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \end{aligned}$$

Explanations:
x_t: input tensor at t
h_t: hidden output tensor at t
W, U: learnable weights
z_t: update value
r_t: reset value
$\odot$: the Hadamard Product
σ, ϕ: sigmoid and tanh activation functions
$\hat{h_t}$: candidate hidden output tensor at t
h_t: final hidden output tensor combined with previously hidden output

Convolutional GRUs (CGRU) can be used for sequential image processing and contain almost the same series of operations, except now the hidden state is concatenated to the current input on which learned kernels are convoluted. The *tanh* activation function is also replaced by a rectified linear unit (ReLU) function.

3.4.6. 3D Convolutional Neural Networks

Whereas CNNs are typically used to process 2-dimensional data one by one, a 3D CNN [13] allows for an extra dimension, for example, a temporal dimension for video [14], or a depth dimension for depth estimation [6]. This is made possible by increasing the dimensionality of the kernel, to 3D, which means that the kernel now slides in the width, height, and depth of the volume. While 3D CNNs are able to process more complex data, they cause significantly more overhead in terms of computational complexity and memory. Given a filter with dimensions H_f, W_f, and for 3D D_f, and input dimensions H_{in}, W_{in}, and possibly D_{in}, we can relate the memory and computational complexity for a single convolution as follows:

2D Convolution

- **Computational Complexity**: $H_{in} \times W_{in} \times H_f \times W_f \times C_{in} \times C_{out}$
- **Memory Complexity**: $H_f \times W_f \times C_{in} \times C_{out}$

3D Convolution

- **Computational Complexity**: $H_{in} \times W_{in} \times D_{in} \times H_f \times W_f \times D_f \times C_{in} \times C_{out}$
- **Memory Complexity**: $H_f \times W_f \times D_f \times C_{in} \times C_{out}$

3.4.7. Transformers

While recurrent neural networks (RNNs) allow the processing of sequential information, they are inefficient to train due to their sequential input processing and lack of contextual relationships between inputs. The transformer architecture [1] solves both of these problems through a completely different approach: the self-attention mechanism. The success and wide adoption of the transformer can partly be explained by its flexibility to accept any input, even images, making it one step closer to a general-purpose model. Transformers seem to outperform most older architectures on natural language processing (NLP) and computer vision tasks, but this comes at the cost of needing more data and more compute. This makes this type of architecture mostly interesting to big tech corporations who have access to this kind of data and compute. Nevertheless, the effectiveness of transformers keeps surprising researchers and users, and the big question is where it will take us in the future.

Self-Attention

We will now explain the basic principles behind self-attention. First, the sequential input is tokenized, meaning broken down and encoded in a numerical vector with a fixed size. When the goal is processing text, sentences can be broken down into words and then converted to vectors. A popular method for doing this is Word2Vec [15], which encodes words in such a way that similar or related words will also have higher similarity in the numerical representation. When the input is an image, it can be broken down into ”patches” and fed into a linear layer to produce a numerical token.

The idea behind attention is to determine for every token, how much attention it should pay to every other token. For example, in the sentence ”I let the dog out”, the token ”I” should probably pay more attention to ”dog” than to ”the”. Basically, self-attention reweighs the tokens by analyzing the context, thereby producing a more meaningful encoding of the input sequence (rather than simply encoding individual words). The original authors of the transformer model describe this concept more formally using keys, queries and values, which could be described using the following analogy:

- **Keys**: what characteristics does a token possess?
- **Queries**: what information is this token looking for?
- **Values**: what is the actual value or contribution of this token to the sequence?

The key (K), query (Q) and value (V) are obtained by passing the input tokens through linear layers. The formula for calculating the attention is as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k T}})V$$

Here, d_k denotes the dimensionality of the key vector and is used to reduce the magnitude of the attention and stabilize the gradients. T is often included as an additional hyperparameter and stands for the *temperature* of the transformer. A higher temperature will ensure that attention is spread out more, whereas a lower temperature will focus the attention on the highly activated fields.

Multi-Head Attention

To make the transformer network more versatile, the original authors also proposed adding multi-head attention. This allows the network to learn multiple representations of a sequence, which can be seen as looking at the input from different perspectives. Multi-head attention can be implemented using the following formulas:

$$\text{MultiHeadAttention} = \text{Concat}(\text{head}_1, \ldots, \text{head}_n)W^O$$
$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Positional Encoding

The standard attention mechanism ignores the position or order of the tokens. This can be critical information, for example, if you compare the sentences ”the cat bit the dog” or ”the dog bit the cat”. With positional encoding, the tokens are augmented with position-dependent values such that a token value not only depends on the original input but also on the location in the sequence. The original authors propose the following positional encoding:

$$\text{PE}(pos, 2i) = sin(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

$$\text{PE}(pos, 2i+1) = cos(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

Here, i denotes the index of the dimension and d_{model} the total token dimension.

Classic Vision Transformers

Originally developed for encoding language, transformers turn out to be surprisingly effective on images as well. Vision Transformers (ViTs) [2] tokenize input images and process them in the exact same way as normal tokens. Tokenizing 2D images is done in two steps:

1. Patching: the image is divided into equal squares or ”patches”.
2. Tokenization: every patch is fed through a small CNN, after which the output channels are flattened and fed through a linear layer to obtain the final embedding dimensions.

Typically, ViTs require a lot more data and training time compared to CNNs but are able to achieve higher accuracy on common benchmarks like ImageNet. The general formatting of a token also allows the development of multimodal models that can for example accept images and text simultaneously.

Hierarchical Vision Transformers

While ViTs have proven to be a superior backbone for downstream tasks compared to older CNN-based backbones, calculating the self- and cross-attention is computationally expensive, as it grows quadratically with the number of pixels in the input image. The Swin Transformer [16] reduces the amount of computations by only calculating the self-attention on local regions (windows) in the image. Because global self-attention is still desired, these local ”windows” of attention are merged into a single patch, after which the process is repeated this time on a more global scale. This approach has similarities to a Feature-Pyramid Network (FPN), where the receptive field is also gradually increased from local to global. Swin also introduces the shifted windows technique, which shifts the window range such that patches are also included in other windows. Shifting results in empty spots, which are filled with the patches that end up outside the window range.

Pyramid Vision Transformers

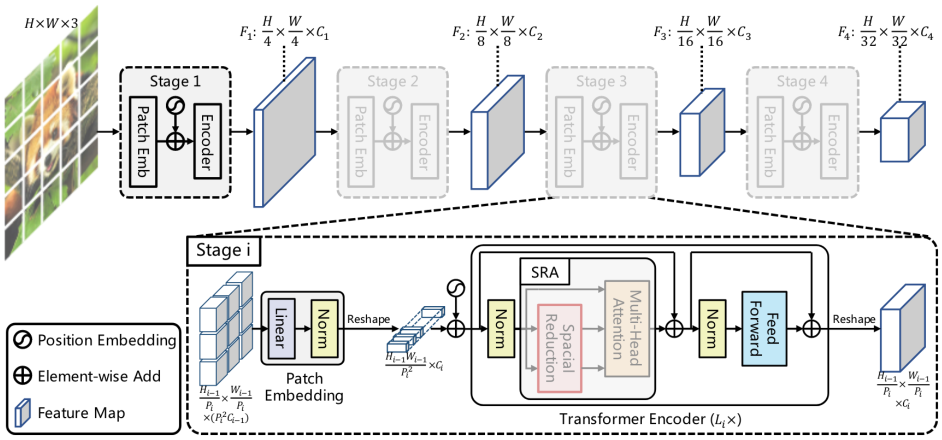

$H \times W \times 3$
$F_1: \frac{H}{4} \times \frac{W}{4} \times C_1$
$F_2: \frac{H}{8} \times \frac{W}{8} \times C_2$
$F_3: \frac{H}{16} \times \frac{W}{16} \times C_3$
$F_4: \frac{H}{32} \times \frac{W}{32} \times C_4$
Stage 1
Patch Emb
Encoder
Stage 2
Stage 3
Stage 4
Stage i
Linear
Norm
Patch Embedding
Reshape
SRA
Spacial Reduction
Multi-Head Attention
Norm
Feed Forward
Transformer Encoder ($L_i \times$)
Position Embedding
Element-wise Add
Feature Map

Figure 3.10: Overall architecture of the Pyramid Vision Transformer (PVT). *Reprinted from [17, Fig. 3]*

Pyramid Vision Transformers (PVT) [17] bear a lot of similarities to the CNN-based Feature Pyramid Networks (FPNs), in the sense that the receptive field is gradually increased and that the network can serve as a multi-level backbone. Note how in every block, tokens are converted into patches again, resulting in a representation that is useful for downstream dense prediction tasks.

Twins: Efficient Pyramid Vision Transformers
The authors of ”Twins: Revisiting the Design of Spatial Attention in Vision Transformers” [18] introduce two new variants of the ViT: Swins-PCPVT and Swins-SVT. Both models are an iteration on the PVT.

- Twins-PCPVT: the authors noted that the default positional encoding technique used in PVT is suboptimal, as the authors of CPVT [19] explain by introducing the Positional Encoding Generator (PEG). The main issue with standard positional encoding is that it reduces the flexibility of the model, as it causes difficulties with varying input sizes once trained. The PEG solves this by including a 2D convolution block over the patch embeddings, providing the network with a learnable positional encoder. Experiments by the authors of Twins show that replacing the standard PE with PEG gives a significant boost to the precision on the COCO dataset.
- Twins-SVT: as the PVT only uses the computationally expensive global attention, applying it in the wild requires a large amount of computational resources. To remedy this, the Twin-SVT uses local self-attention (LSA) and global self-attention (GSA) with PVTs. This technique allows the network to exchange information between patch groups, whilst being much more efficient. Twins-SVT also uses PEG, which makes it the more favorable option to use.

Both models have a pretrained version published online based on the Ade20k dataset, which opens up the possibility to serve as a feature extractor without the need for expensive training.

3.5. Deep-Learning-Based Multi-View Stereo

3.5.1. Introduction

Multi-View Stereo (MVS) has seen a significant paradigm shift since the rise of using deep learning for computer vision. Traditional methods, like SfM, were mainly relying on explicit formulations of geometric and photometric consistency, which resulted in accurate, but sparse estimations of a scene, whilst also requiring a significant amount of input images for decent results. Attempts to fill in the gaps were mainly limited by textureless regions, occlusions and view-dependent effects making it difficult to find pixel-to-pixel correspondences.

The use of deep learning in the field of MVS has opened up the possibility to deal with these problems, through the ability of neural networks to understand the context of a scene and image in order to make predictions. In this section, we highlight the main mechanics behind common deep-learning-based MVS methods and discuss the challenges and limitations.

3.5.2. Typical MVS Pipeline

Modern MVS depth prediction methods select a *reference view* and a set of *source views* that are used for triangulating and matching features with the reference view, which often happens pair-wise. A typical pipeline for recent MVS depth estimation methods is depicted in Fig. 3.11. Essentially, these methods can be broken down into four stages:

- **Feature extraction**: This stage is responsible for extracting features that are representative of the surface. Ideally, the features maximize the similarity of identical surface points between views and minimize similarity on all points. Typically, a Feature Pyramid Network (FPN) is used to extract features at various receptive field sizes, while embedding the low-level features with the high-level understanding of the scene. This also allows for a coarse-to-fine approach in the refinement stage.
- **Warping**: This stage uses the typical, explicit geometrical formulation of the pinhole camera model. Since the depth is unknown at this stage, a series of fronto-parallel planes are formed for the reference camera at discrete, predetermined distances. The planes form the hypothesis range for the depth estimation, so need to be configured correctly depending on the scene and camera setup. The features from the source images are warped onto these planes, using epipolar geometry explained in previous chapters. A common method to fuse the source and reference features is to take the dot product, which is an indicator of feature similarity. This is then fed through a *Softmax* along the depth axis and added to the aggregate volume. This data structure is often called a cost volume.
- **Regularization**: The recent successes in deep-learning-based MVS can be largely attributed to the regularization stage, which is able to deal with the typical problems (textureless, occlusions, view-dependent effects) by predicting the unknowns based on the 3D patterns in the cost volume. A 3D CNN is able to regularize and smoothen the predictions. Finally, a confidence estimation per x, y coordinate along the depth-axis is obtained by taking the *Softmax* along the depth.
- **Refinement**: Using the obtained probability distribution, the network can predict a per-pixel depth value using either the index of the highest confidence or the *Soft-Argmin*, which is the weighted average index of all probabilities. A key advantage of using *Soft-Argmin* is that it is differentiable, facilitating efficient backpropagation of the loss.

3.5.3. Loss Functions

The output of a typical dl-based MVS method is a pixel-wise confidence estimation along the depth hypotheses. How this estimation is used can differ per method, but broadly speaking can be categorized into regression-based and classification-based estimation.

Regression-based depth estimation predicts a continuous depth value for every pixel, for example using the *Soft-Argmin* function. Typical loss functions for this type are Mean Squared Error (MSE), Mean Absolute Error (MAE) or Huber loss.

Classification-based estimations are predictions based on a discrete, binned range of predetermined hypotheses. Typical loss functions here are Cross-Entropy (CE) loss, focal loss or Wasserstein loss.

The authors of MVSFormer [6] note that regression-based models can be overconfident, even for out-of-range depth hypotheses, but seem to be able to produce more accurate point clouds. The confidence

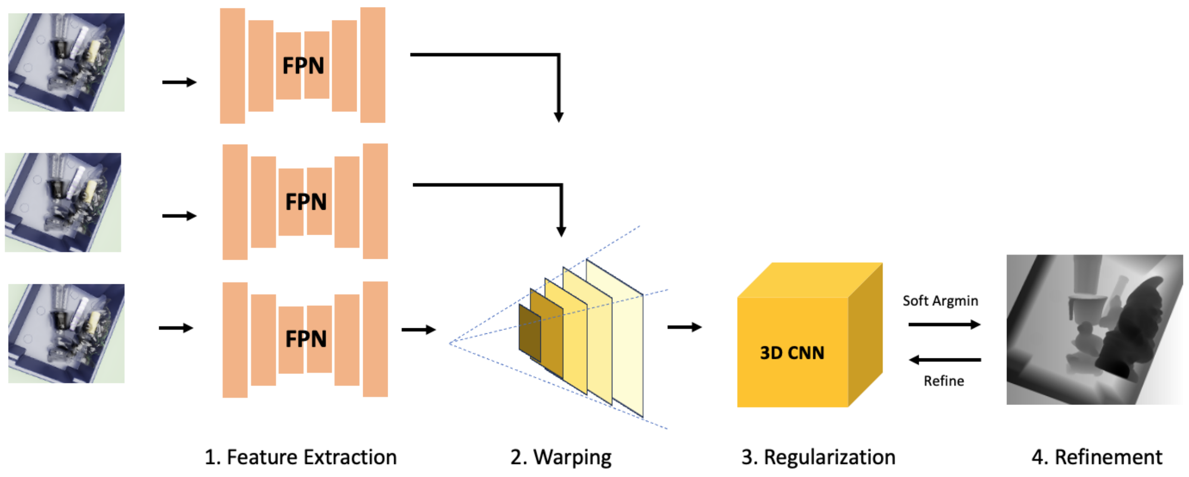


Figure 3.11: Visualization of a typical deep-learning-based MVS pipeline (simplified).

maps from the classification-based experiment were better; which is especially useful for filtering out predictions that suffer from occlusion or strong view-dependent effects.

3.5.4. Challenges and Limitations

Recent advancements in deep-learning-based MVS have shown that we can build systems that deal with the typical problems that MVS suffers from. Still, these problems are far from solved, and challenges remain for mitigating the effects. In this subsection, we list some of the current limitations and challenges we think are in front of us.

- **Large dataset requirements**: current MVS methods require vast amounts of data in order to perform well enough to be used reliably within the context of the data itself. It is an open question (with deep learning in general) whether we can unlock methods that are able to learn from far less data, like humans can. Creating datasets is expensive, partly because it requires ground-truth depth data, which is costly to obtain for real-world settings. Modern synthetic data generation pipelines can help us overcome this hurdle.
- **Generalizability**: an ideal MVS system would work in any setting reliably, meaning that the model generalizes well from the data it was trained on. Whereas SfM generalizes to any setting, deep-learning-based MVS is not guaranteed to work on settings that it was not trained on.
- **Computational complexity**: Deep-learning-based MVS can be expensive to train, often taking days on costly hardware. Even though systems are able to achieve near real-time inference, the model complexity would ideally be decreased to allow running it on cheaper hardware.

3.6. MVS Datasets and Benchmarks

A benchmark contains a publicly available dataset that is used to assess the performance of new models and compare them to previous state-of-the-art. Since MVS is quite a broad topic with many applications, there are many different benchmarks each with its own niche, such as the type of scenes, number of cameras, and lighting conditions. In this section, we highlight a few of the most commonly used benchmarks on the topic of MVS surface reconstruction.

3.6.1. Common MVS Benchmarks

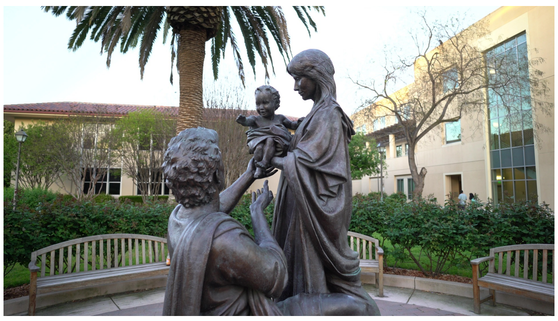


Figure 3.12: 'Family' scene in T&T [20]

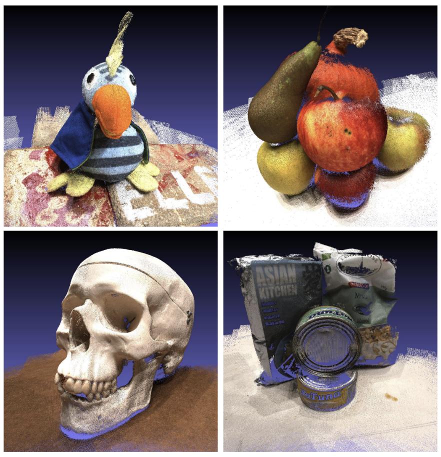


Figure 3.13: Subset of point clouds in DTU [21]

Tanks&Temples
The Tanks&Temples [20] dataset contains a mix of 14 test and 7 training indoor and outdoor scenes with on average more than 10000 images per scene. Depth information was captured using an industrial laser scanner. The goal of this benchmark is to stimulate the development of a model that can reconstruct a scene in 3D from handheld camera footage. At the time of writing, MVSFormer [6] is at the top of the leaderboard with a mean F-score of 66.41.

DTU
The DTU dataset [21] is nearly always used as a benchmark in MVS works, making it a solid reference point to compare results against multiple popular methods. When DTU was released, the aim was to serve as a dataset with more variation than was previously available. The set has 124 real scenes, with 49 or 64 camera positions, with accurate ground-truth depth and 7 different lighting conditions. See Fig. 3.13 for samples. Even though the dataset is relatively old, it remains an invaluable resource for MVS related research. The data was captured in a controlled environment, with the cameras focused at the center of the scene, making it a useful dataset for analysing fine-grained differences between results.

ETH3D
The ETH3D [22] dataset was put together by ETH Zurich and contains images from real scenes with ground-truth depth (see Fig. 3.14. Compared to DTU, this dataset contains significantly more occlusion and textureless surfaces, making it a more advanced evaluation.

Middlebury
The Middlebury dataset [23] contains challenging scenes with cluttered objects, again with a lot of occlusions, textureless and reflective surfaces. The data only contains one stereo pair of images and ground-truth disparity per scene (Fig. 3.15), which makes it incompatible with 3D scene reconstruction tasks. Even though the dataset was released in 2002, researchers are still actively releasing new and improved versions, with the latest one being released in 2021.

BlendedMVS
BlendedMVS [24] strongly focuses on 3D reconstruction, by generating a training set based on the reconstructed 3D mesh from the original images, instead of being based on expensive laser scanners. In

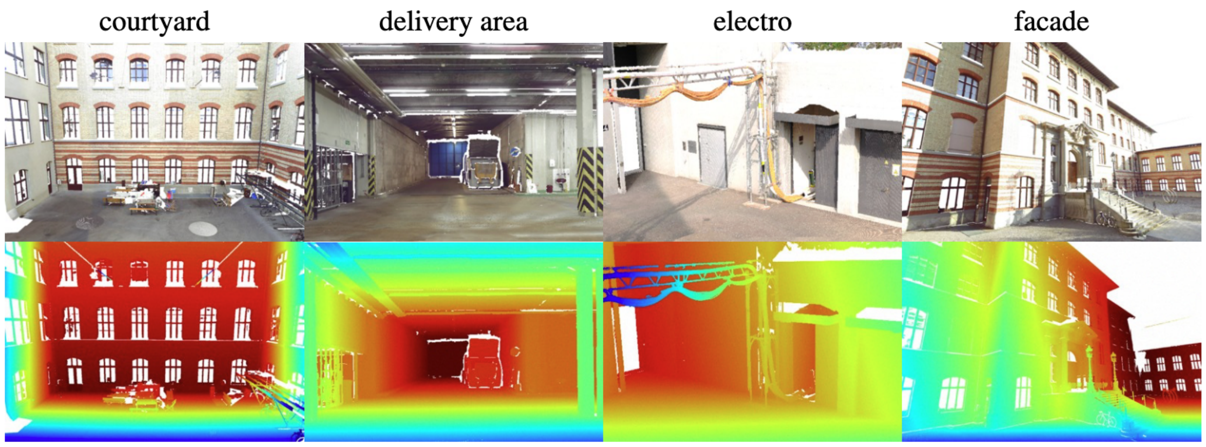


Figure 3.14: Sample scenes from ETH3D [22]

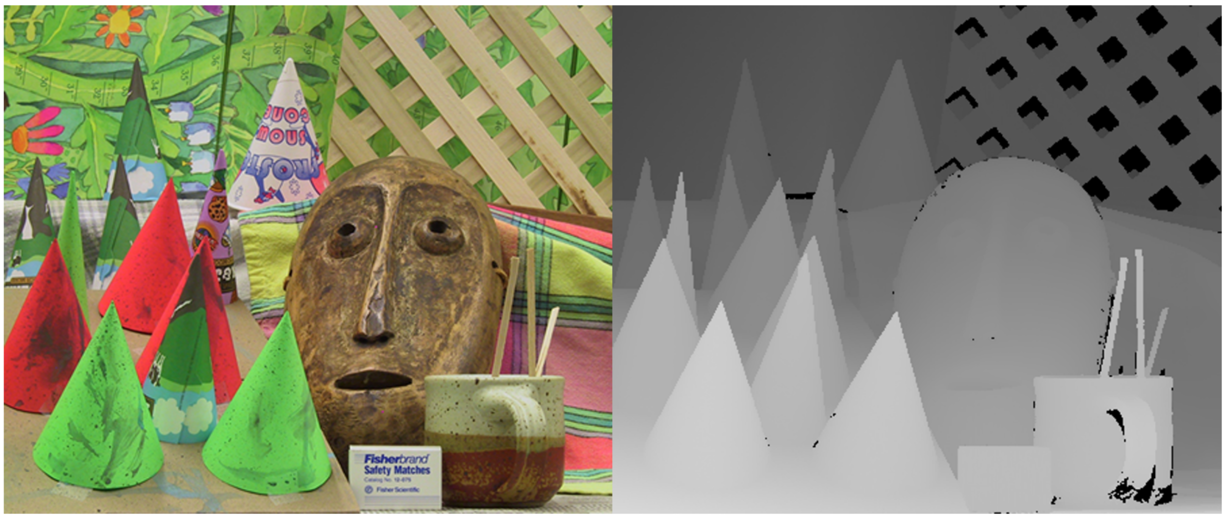


Figure 3.15: Sample scene from the Middlebury dataset with ground-truth disparity. [23]

order to still have view-dependent effects, these reconstructions are *blended* with the original input image, thereby embedding the 2D projection with ambient lighting. According to the authors, training on this data results in improved scores on T&T compared to training on DTU or ETH3D, which can partly be explained by the fact that BlendedMVS contains significantly more ground-truth depth.

3.6.2. Transparent MVS Datasets

ClearPose

The authors of ClearPose [25] recognized the lack of large-scale RGBD datasets focusing on transparent objects. To this extent, they built a dataset with more than 350K images and 5M instance annotations of transparent objects. Ground-truth depth was annotated by combining laser scanner depth with digital models, allowing for a complete depth view and simplified extraction of instance masks.

Trans10K

Trans10K [26] is a real RGB dataset with images of segmented transparent objects in the wild, which can serve as a valuable resource for training models to detect things like windows. For MVS, this dataset falls short, due to the lack of multi-view imagery.

Syn-TODD

The Toronto Transparent Object Depth Dataset (Syn-TODD) was published as part of the MVTrans [27] work, which focused on training for robotic manipulation in household or laboratory settings. While Syn-TODD is rich in terms of feature count, it does only contain transparent objects, making it difficult to evaluate relative performance to other material types.

References

[1] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[2] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *ICLR* (2021).

[3] Muhammad Muzammal Naseer et al. “Intriguing properties of vision transformers”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 23296–23308.

[4] Robert Geirhos et al. “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.” In: *International Conference on Learning Representations*. 2019. URL: `https://openreview.net/forum?id=Bygh9j09KX`.

[5] Sayak Paul et al. “Vision transformers are robust learners”. In: *Proceedings of the AAAI conference on Artificial Intelligence*. Vol. 36. 2. 2022, pp. 2071–2081.

[6] Chenjie Cao et al. “MVSFormer: Multi-View Stereo by Learning Robust Image Features and Temperature-based Depth”. In: *Transactions of Machine Learning Research* (2023).

[7] H Christopher Longuet-Higgins. “A computer algorithm for reconstructing a scene from two projections”. In: *Nature* 293.5828 (1981), pp. 133–135.

[8] Zhengyou Zhang. “A flexible new technique for camera calibration”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334.

[9] Johannes Lutz Schönberger et al. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[10] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[11] Yann LeCun et al. “Handwritten digit recognition with a back-propagation network”. In: *Advances in neural information processing systems* 2 (1989).

[12] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[13] Shuiwang Ji et al. “3D convolutional neural networks for human action recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 221–231.

[14] Du Tran et al. “Learning spatiotemporal features with 3d convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[15] Kenneth Ward Church. “Word2Vec”. In: *Natural Language Engineering* 23.1 (2017), pp. 155–162.

[16] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.

[17] Wenhai Wang et al. “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 568–578.

[18] Xiangxiang Chu et al. “Twins: Revisiting the design of spatial attention in vision transformers”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 9355–9366.

[19] Xiangxiang Chu et al. “Conditional Positional Encodings for Vision Transformers”. In: *ICLR 2023*. 2023. URL: `https://openreview.net/forum?id=3KWnuT-R1bh`.

[20] Arno Knapitsch et al. “Tanks and temples: Benchmarking large-scale scene reconstruction”. In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), pp. 1–13.

[21] Henrik Aanæs et al. “Large-scale data for multiple-view stereopsis”. In: *International Journal of Computer Vision* 120 (2016), pp. 153–168.

[22] Thomas Schops et al. “A multi-view stereo benchmark with high-resolution images and multi-camera videos”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3260–3269.

[23] Daniel Scharstein et al. “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms”. In: *International journal of computer vision* 47 (2002), pp. 7–42.

[24] Yao Yao et al. “BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks”. In: *Computer Vision and Pattern Recognition (CVPR)* (2020).

[25] Xiaotong Chen et al. “ClearPose: Large-scale Transparent Object Dataset and Benchmark”. In: *European Conference on Computer Vision*. 2022.

[26] Enze Xie et al. “Segmenting Transparent Objects in the Wild”. In: *arXiv preprint arXiv:2003.13948* (2020).

[27] Yi Ru Wang et al. *MVTrans: Multi-View Perception of Transparent Objects*. 2023. arXiv: `2302.11683 [cs.RO]`.

Transparency and specularity are challenging phenomena that modern depth perception systems have to deal with in order to be used in practice. A promising family of depth estimation methods is Multi-View Stereo (MVS), which combines multiple RGB images to predict depth, thus circumventing the need for costly specialized hardware. Although promising, finding pixel-to-pixel mappings between images is a challenging task, clouded by ambiguity. In order to determine the current ability to deal with such ambiguity, we introduce ToteMVS: a multi-view, multi-material synthetic dataset with diffuse, specular and transparent objects. Recent works in computer vision have effectively replaced Convolutional Neural Networks (CNNs) with the emerging Vision Transformer (ViT) architecture, but it remains unclear whether ViTs outperform CNNs in handling reflective and transparent materials. In our study, we use ToteMVS to compare ViT- and CNN-based architectures on the ability to extract useful features for depth estimation on diffuse, specular, and transparent objects. Our results show that, in contrast with the current trend of using ViTs over CNNs, the ViT-based model does not have a special capability for dealing with these challenging materials in the context of MVS.