# Self-supervised recurrent depth estimation with attention mechanisms

**Ilya Makarov**[1,2]**, Maria Bakhanova**[2]**, Sergey Nikolenko**[3,4]**, and Olga Gerasimova**[2]

[1] **Artificial Intelligence Research Institute (AIRI), Moscow, Russia**
[2] **HSE University, Moscow, Russia**
[3] **Steklov Institute of Mathematics at St. Petersburg, St. Petersburg, Russia**
[4] **St. Petersburg State University, St. Petersburg, Russia**

Corresponding author:
First and second authors contributed equally to this work.
Corresponding authors are the first and fourth authors.[1]

Email address: iamakarov@hse.ru,ogerasimova@hse.ru

## ABSTRACT

Depth estimation has been an essential task for many computer vision applications, especially in autonomous driving, where safety is paramount. Depth can be estimated not only with traditional supervised learning but also via a self-supervised approach that relies on camera motion and does not require ground truth depth maps. Recently, major improvements have been introduced to make self-supervised depth prediction more precise. However, most existing approaches still focus on single-frame depth estimation, even in the self-supervised setting. Since most methods can operate with frame sequences, we believe that the quality of current models can be significantly improved with the help of information about previous frames. In this work, we study different ways of integrating recurrent blocks and attention mechanisms into a common self-supervised depth estimation pipeline. We propose a set of modifications that utilize temporal information from previous frames and provide new neural network architectures for monocular depth estimation in a self-supervised manner. Our experiments show that the proposed method effectively exploits temporal information for real-time depth estimation tasks.

**Keywords:** Computer Vision, Depth Reconstruction, Autonomous Vehicles, Augmented Reality, Deep Convolutional Neural Networks, Recurrent Neural Networks, Self-Supervised Learning, Attention Mechanism

## 1 INTRODUCTION

Depth estimation is an essential basic problem for a variety of computer vision applications. It is key to understanding 3D scene geometry and can provide essential cues in many tasks, including object detection (Gupta et al., 2014), scene reconstruction (Menze et al., 2015, 2018; Shin et al., 2019), simultaneous localization and mapping (SLAM) (Yang et al., 2018), image classification (He, 2017), and others. In practice, depth maps can be obtained directly, e.g., with the help of light detection and ranging (LiDAR) devices. Although LiDARs are widely used in practice, they produce low-resolution results, and their perception distance is short. Modern methods of depth estimation from image sequences do not suffer from these limitations, and due to the growing success and generally excellent performance of modern computer vision models and algorithms they can often supplement or even replace hardware sensors. In this work, we consider the problem of depth estimation through joint optimization of scene structure and camera motion across sequences of RGB images. This approach is called *Structure from Motion* (SfM) (Schönberger and Frahm, 2016), and it implies training in a self-supervised manner.

Over the past decade, depth estimation has been considered in numerous studies. Many theoretical and conceptual frameworks have been put forward to boost depth prediction quality (Eigen et al., 2014; Laina et al., 2016; Eigen and Fergus, 2014; Fu et al., 2018). Recent improvements in self-supervised depth estimation have also resulted in important performance improvements (Zhou et al., 2017; Godard et al., 2019; Guizilini et al., 2020a). We especially note methods that combined both approaches: depth prediction and information produced by a LiDAR sensor or SLAM (Ma and Karaman, 2017; Ma et al., 2018).

Despite the progress of modern depth estimation methods, a number of challenging problems still remain. One key problem is the fact that depth estimation is ill-posed: the same input image can project to different depths. Moreover, existing depth estimation models are often very sensitive to changing external conditions such as lighting and noise. Self-supervised approaches to depth estimation rely on ego-motion information and do not take into account ground truth depth maps, which implies that these methods usually perform worse than supervised ones.

Existing approaches, both supervised and self-supervised, usually estimate depth based on a single image, although in practice it is very often possible to utilize information from image sequences. There do exist state-of-the-art self-supervised methods that already use frame sequences for camera pose estimation and view synthesis further down the line (Zhou et al., 2017; Godard et al., 2019). However, most of them still do not utilize this temporal information in depth prediction. We believe that the performance of current state-of-the-art methods can be further improved with temporal knowledge. To achieve this, we aim to provide new modifications that can potentially increase depth accuracy in the monocular setting from an image sequence.

In this study, we focus on self-supervised depth estimation. We propose a modified architecture that can utilize temporal information in a depth model, integrating recurrent ConvGRU and Fusion blocks with self-attention into existing self-supervised training pipelines. To summarize, our main contributions are as follows:

1. We propose a set of modifications that result in new network architectures that utilize temporal information across frame sequences. These modifications are based on recurrent blocks and fusion of their hidden states.

2. We design a training strategy for the recurrent self-supervised depth prediction task.

3. We conduct experiments and provide results of the impact of our modifications. We provide an ablation study that helps to estimate the effect of each component that we propose. Our experiments show that the proposed methods can be an effective tool for exploiting temporal information for depth estimation in real time.

The code for reproducing experiments is also released accompanying our paper[1]. The paper is structured as follows. Section 2 gives an overview of the current methods in depth estimation. In Section 3, we provide a theoretical basis for our experiments. Section 4 describes our experimental setup, Section 5 provides their results, and in Section 6 we provide visual examples and their analysis. Section 7 summarizes and concludes the paper.

---

[1]https://github.com/MariBax/self-supervised-depth-estimation

## 2 RELATED WORK

In this section, we provide an overview of existing depth estimation methods. We begin with supervised models that directly learn to predict depth maps from single RGB images. Next, we address the problem of self-supervised learning and SfM-based methods. Finally, we proceed to existing self-supervised approaches which are most relevant to our work.

### 2.1 Supervised depth estimation

The idea of supervised depth estimation implies direct learning of a hidden mapping between the input image and its depth. Usually, this approach includes regression-based methods because of the continuous nature of depth maps.

One of the earliest applications of deep learning to this problem (Eigen et al., 2014) introduced a network that consists of two components: one estimates the global scene structure, and the other refines the results using local information. In the stereo-based approach, Xie et al. (2016) designed a convolution neural network that generates a corresponding right view of a stereo pair by combining information from multiple levels. Disparity distribution for pixels is optimized with supervised photometric loss. Cao et al. (2016) reformulated the supervised depth estimation task as a pixel-wise classification problem that was solved with a fully convolutional deep residual network.

Supervised methods mostly rely on convolutional architectures. One important difference between methods is that they tend to utilize a large variety of objective functions that define the difference between the ground truth depth map and its prediction. The most straightforward approach is to use $L_1$ or $L_2$ distances. Reverse Huber loss (Laina et al., 2016) is a popular example of a combination of these two distances. To address the scale problem, some studies focused on the loss as a distance between predicted and real depth in log space (Eigen et al., 2014; Eigen and Fergus, 2014). Yin et al. (2019) argued that most metrics neglect geometric constraints in the three-dimensional space, so they developed a loss that enforces geometrical constraints. Another alternative to pixel-wise metrics is perceptual loss (Johnson et al., 2016), which was successfully used in many depth-related tasks ((Makarov et al., 2017a,b, 2018a; Makarov and Korinevskaya, 2019; Makarov et al., 2018b,c, 2019; Maslov and Makarov, 2021)). To reduce blurring of estimated depths, a new model with fusion mechanisms was proposed by Hu et al. (2019). It consists of a module that can fuse multi-scale information produced by an encoder, a refine module to fuse decoder outputs and multi-scale fusion modules. In their objective function, the authors combined the difference in gradients and surface normals with balanced Euclidean loss in log space.

State of the art performance in the supervised setting has been achieved by Fu et al. (2018), who proposed the SID policy and ordinal regression loss. Despite the excellent performance of the model, its main disadvantage is a huge number of parameters that leads not only to long training time but also to large inference times in practice. Therefore, a challenge arises to build a lightweight model that is able to process frame sequences in real-time with reasonable quality. Recently, Maslov and Makarov (2020) proposed an architecture that appears to be suitable for real-time robotic applications. The model utilizes temporal information from image sequences with the help of a convolutional gated recurrent unit (ConvGRU) and convolutional long short-term memory (ConvLSTM). The authors showed that these recurrent blocks can lead to improvements in depth estimation accuracy.

### 2.2 Self-supervised depth estimation

As deep learning models for computer vision become larger and more expressive, the data requirements for training them rapidly grow as well. Creating a labeled dataset for depth estimation by hand or with special hardware is extremely laborious and expensive. One solution would be to use synthetic data (Nikolenko, 2021) that has indeed been extensively used for depth estimation. An alternative approach would be to learn from unlabeled monocular sequences or stereo image pairs, as has been done, e.g., in (Garg et al., 2016; Godard et al., 2016). This approach does not utilize ground truth depth maps. One of the first models to perform self-supervised depth estimation (Garg et al., 2016) consists of a convolutional neural network that estimates depth maps from paired stereo images. A predicted depth map is used in the geometrical transformation of the left image into the synthesized right image or vice versa. The loss function here consists of the difference between the synthesized and real images, defined as the structural similarity index measure (Wang et al., 2004), and depth regularization terms.

Further studies mostly focused on generalizations of this approach in the monocular setting. Monocular video-based methods usually utilize consecutive frames to simultaneously learn the depth and pose. Given

a target image and adjacent frames as input, a model produces the depth for the target image and estimates relative poses between the target image and nearby images. Next, the target view is reconstructed with the predicted depth map and relative poses. The main difference between stereo-based and monocular video-based approaches is that poses for stereo data are known and fixed, while in the monocular setting, they also need to be predicted.

Most state-of-the-art methods in self-supervised depth estimation are based on encoder-decoder architectures. Next, we highlight a few such models that are the most relevant for our study. *Monodepth2* (Godard et al., 2019) has been probably the most influential approach in self-supervised depth estimation because of its simplicity. Its architecture consists of a depth net and a pose net; both nets consist of a ResNet encoder and a decoder. The main contribution of *Monodepth2* is a set of improvements that together lead to superior results in depth estimation. The authors introduced the minimum reprojection loss as a substitute for the average reprojection loss and showed that the new loss is more effective in handling occlusions. In addition, Godard et al. (2019) proposed a multi-scale pyramidal upsampling method and auto-masking loss that reduced visual inconsistency. As a result, the model showed high-quality results on the KITTI benchmark, and since then has become a starting point for many researchers. In this work, we mostly follow *Monodepth2* architecture and propose a set of additional components that can be used to extract information from previous frames.

Guizilini et al. (2020a) recently proposed a new architecture called *PackNet-SfM*. Its key feature is the design of packing and unpacking blocks that are considered to preserve the 3D geometrical structure of depth maps. Moreover, the authors presented a new velocity supervision loss that solves the problem of scale ambiguity. Depth scale ambiguity implies that predicted depth maps are usually scaled with predefined constants. In PackNet, camera velocity is estimated and used to compute the loss function that helps reduce the problem with scaling. PackNet has achieved excellent performance on KITTI, but it utilizes heavy 3D convolutions, resulting in a long processing time.

In this work, we intend to investigate the effect of temporal information in depth estimation in the self-supervised setting. Despite the fact that most existing models do not focus on capturing temporal knowledge, there is a number of novel models that try to follow a multi-frame or sequential approach in depth estimation (Watson et al., 2021; Hur and Roth, 2021; Kuznietsov et al., 2021). In particular, Watson et al. (2021) proposed to utilize nearby frames if they are available at test time and designed a model called *ManyDepth*. Its key feature is a cost volume block that helps overcome the scale ambiguity that arises from self-supervised training. Experiments on KITTI and *Cityscapes* show that *ManyDepth* outperforms most existing self-supervised approaches. Hur and Roth (2021) focused on the scene flow task where 3D scene flow and depth are estimated jointly in a self-supervised setting. They propose a multi-frame model that uses triplets of frames and recurrent blocks, and it shows state-of-the-art performance for the scene flow task.

## 3 METHODS

The self-supervised setting implies that ground truth depth maps are not available during training. The standard approach here would be to train the model using input image reconstruction as supervision. In this case, the model takes as input a list of monocular images or stereo pairs, where one of the images is called the target image. By estimating the depth in the target image and projecting it to nearby views, we reconstruct the target image and try to minimize the reconstruction error.

In this section, we describe our model in detail. First, we review a common approach to self-supervised depth estimation. We provide details about network structure and objective functions. Next, we describe the components of our network: recurrent block ConvGRU, self-attention layer, and the fusion technique.

### 3.1 Problem Formulation

In the self-supervised monocular setting, we operate with frame sequences extracted from videos. Given these consecutive frames, we predict the target view from the viewpoint of another image. In order to synthesize the target image, we use a hidden variable, the depth map, and estimated relative pose between the target image and nearby view. The overall self-supervised pipeline is illustrated in Figure 1.

Formally, the problem can be described as follows. Suppose that we are given $N$ adjacent images that can be considered as a frame sequence $\{I_1, \ldots, I_N\}$. Usually, one image $I_t$ is chosen as the target image, while other frames $S = \{I_s\}_{s=1, \ s\neq t}^{N}$ are considered as context. This sequence is an input for our model, which consists of two parts:
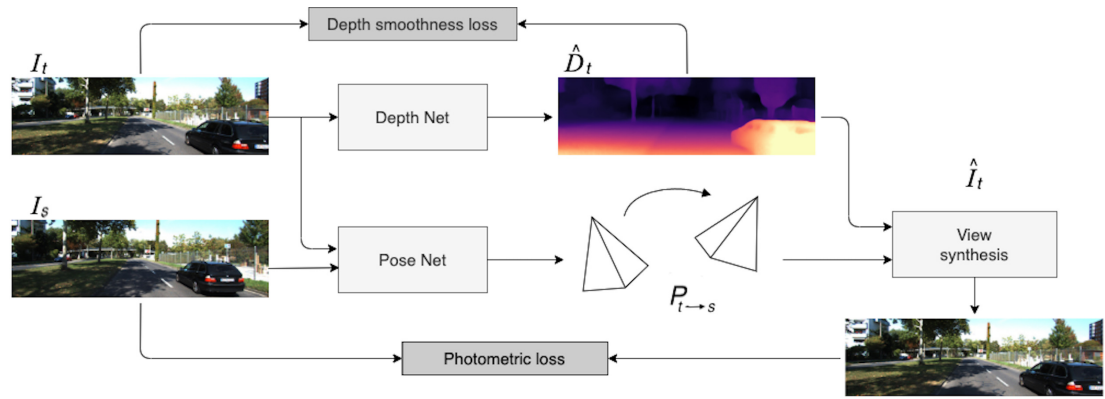
**Figure 1.** Self-supervised monocular SfM network architecture. First, the target image $I_t$ is processed by *Depth Net*. At the same time, target and context frames $(I_t, I_s)$ are passed through *Pose Net*. Then, using 6-DoF relative pose $P_{t \to s}$ and depth map $\hat{D}_t$, the target image $\hat{I}_t$ is reconstructed via the inverse warping transformation. The loss consists of two parts: photometric reprojection error between the real target image and the synthesized one and a depth smoothness term. Road images and ground truth depth maps taken from (Menze and Geiger, 2015).

- the depth network (*Depth Net* in Fig. 1) $f_D : I \to D$ that predicts scale-ambiguous depth $D = f_D(I(p))$ for every pixel $p$ in the input image $I$; using this depth network, we estimate the depth for the target image;

- the pose network (*Pose Net* in Fig. 1) $f_P : (I_t, I_s) \to P_{t \to s}$ that estimates a set of 6-DoF relative poses $P_{t \to s}$, $I_s \in S$ between the target image $I_t$ and nearby context frames; for example, if $N = 3$ then $I_t = I_2$ and $I_s = I_i$, where $i \in \{1, 3\}$.

Given a pair $(I_s, I_t)$ consisting of the target image and source image, we can inversely warp $I_t$ to the source frame $I_s$. In order to do this view synthesis, we need to know an estimated depth map $D_t$ for the target image $I_t$ and the transformation $P_{t \to s}$ from $I_t$ to $I_s$. For a given pixel coordinate $p_t$ from $I_t$ which is co-visible in $I_s$, its corresponding pixel coordinate $p_s$ in $I_s$ can be defined by the inverse warping transformation as follows:

$$p_s \sim K_s[P_{t \to s}]D_t(p_t)K_t^{-1}p_t, \tag{1}$$

where $\sim$ denotes equality in the homogeneous coordinates (Shen et al., 2019), and $K_s$ and $K_t$ are intrinsic matrices that are known and fixed. $D_t(p_t)$ is the depth value for a pixel $p_t$, and $[P_{t \to s}]$ is the transformation matrix. Thus, given the depth map and transformation matrix, we can construct a matching function between pixels of the source and target images. This approach can be considered as a geometric projection relationship, and it helps us synthesize views. The difference between a real image and a synthesized one is a key component in the loss function described below.

### 3.2 Objective function

In the objective function, we follow Zhou et al. (2017): the overall self-supervised loss function consists of the photometric reprojection error $L_p$ and depth regularization term $L_s$.

Let $I_t$ be a ground truth target image, and let $I_{s \to t}$ be the target image synthesized from the source image $I_s$. Then the basic photometric reprojection error $L_p$ is defined as

$$L_p = \sum_{I_s \in S} \text{pe}(I_t, I_{s \to t}), \tag{2}$$

where the reprojection error is defined as the weighted combination of the structural similarity index measure (SSIM; see Wang et al. (2004)) and $L_1$ pixelwise difference:

$$\text{pe}(I_a, I_b) = \alpha \frac{(1 - SSIM(I_a, I_b))}{2} + (1 - \alpha)\|I_a - I_b\|_1. \tag{3}$$

208    Following Godard et al. (2019), we set $\alpha = 0.85$.

209    The depth regularization term, first introduced by Godard et al. (2016), helps to regularize the depth
210    in texture-less low-image gradient regions and is defined as

$$L_s(\hat{D}_t) = |\delta_x \hat{D}_t| e^{-|\delta_x I_t|} + |\delta_y \hat{D}_t| e^{-|\delta_y I_t|} \qquad (4)$$

211    As we have mentioned above, we calculate the reprojection loss for a set of source images $I_s$. A
212    common strategy used to compute the final loss is to average the reprojection error from multiple sources,
213    i.e., to average the expression in eq. (2). However, this approach may cause problems with pixels that may
214    be not visible in some source images. Such pixels represent a common problem in the ego-motion study
215    because there is a high probability of encountering objects blocking each other, especially in outdoor
216    scenes. Several works have shown that this problem can be overcome by using masking (Mahjourian et al.,
217    2018; Vijayanarasimhan et al., 2017); however, they do not take into account the recovery of occluded
218    regions. This phenomenon has been carefully studied by Godard et al. (2019), and a simple solution was
219    proposed: the authors replaced averaging with minimum. As a result, the total photometric loss is defined
220    as

$$L_p = \min_{I_s \in S} \mathrm{pe}(I_t, I_{s \to t}). \qquad (5)$$

221    Multi-scale image reconstruction is a common technique in computer vision tasks used to prevent the
222    objective function getting stuck in local minima. Following Godard et al. (2019), we also use multi-scaling
223    in our model. More formally, the photometric loss is defined not just as on the final decoder output but as
224    a combination of loss terms on every scale of the decoder. In this work, we consider 4 scales. Godard
225    et al. (2019) revealed that this approach may result in "holes" in low-texture parts of the image, so Godard
226    et al. (2019) provided a modification for the multi-scaling approach. They upsample depth maps at low
227    scales of the decoder and then compute the photometric loss with these upscaled maps, so that decoder
228    outputs in the loss computation are upscaled to the original image resolution. This approach has proven to
229    be very effective in the self-supervised multi-scale depth estimation pipeline.

230    Self-supervised depth estimation mostly relies on ego-motion. Nevertheless, sometimes motion
231    assumptions may be violated when the camera is in a stationary position but objects move in the scene. As
232    a result, we may have stationary pixels that need to be processed by the model. To overcome these issues,
233    some approaches have been using per-pixel masking (Zhou et al., 2017; Vijayanarasimhan et al., 2017;
234    Luo et al., 2018). The mask is usually incorporated into the loss function, and it can be either predictive,
235    as in (Zhou et al., 2017), or computed by some direct algorithm. In this work, we use the auto-masking
236    method proposed in (Godard et al., 2019). The idea is to track pixels that remain the same between nearby
237    frames, which implies a static position of the camera. The mask is binary, $\mu \in \{0, 1\}$, where the value 0
238    corresponds to pixels where the reprojection error of the synthesized image $I_{s \to t}$ is lower than the error of
239    the original unwrapped image $I_s$:

$$\mu = \left[ \min_{I_s \in S} \mathrm{pe}(I_t, I_{s \to t}) < \min_{I_s \in S} \mathrm{pe}(I_t, I_s) \right]. \qquad (6)$$

**Overall loss function.**    The final loss is computed as a sum of the masked photometric reprojection
term and a smoothness term. We average this result over each pixel and every image in the batch:

$$L = \mu L_p + L_s. \qquad (7)$$

240    In the next sections, we provide an overview of three components that we incorporate into our model
241    to account for multiple frames at the input: recurrent block ConvGRU, self-attention layer, and fusion.
242    Each of the components is essential to our model as will be shown in the ablation study section.

### 3.3  Recurrent block ConvGRU

244    Recurrent neural networks (RNNs) have been widely adopted in the field of computer vision. However,
245    initial RNN models appeared to face difficult training problems, such as exploding or vanishing gradient

effects (Pascanu et al., 2012). A common way to overcome the vanishing gradient problem is to use long short-term memory units, LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000). LSTMs have become one of the most popular approaches in sequence-to-sequence settings, especially in natural language processing (NLP) tasks, including speech recognition (Graves et al., 2013) and machine translation (Parmar and Devi, 2019). By definition, an LSTM block operates with one-dimensional inputs and is not naturally suited to image processing. Nevertheless, Shi et al. (2015) recently proposed a new architecture called ConvLSTM that extends the basic LSTM construction to two-dimensional inputs.

Over the past few years, some studies have used an alternative recurrent mechanism called ConvGRU (Zhang et al., 2019). It was first introduced for the video representation task (Ballas et al., 2015) and has proven to be quite effective in subsequent studies (Siam et al., 2016; Maslov and Makarov, 2020). ConvGRU has a simpler architecture than ConvLSTM while preserving the same gated structure. In this work, we use ConvGRU cells since it has fewer parameters, and it has been proven to be effective in capturing temporal information for online depth estimation (Maslov and Makarov, 2020).

Next, we describe the ConvGRU cell as a particular type of RNN network applied to a sequence of 2D objects with an arbitrary length. ConvGRU can be viewed as a modification of the original GRU cell (Cho et al., 2014). The main difference between these GRU versions is that ConvGRU utilized convolution operations instead of fully connected units used in the original GRU. Each recurrent unit of the ConvGRU cell is designed to capture dependencies on different time scales. Moreover, due to convolution operation, these recurrent units share their parameters across different spatial locations in the input (Ballas et al., 2015). The hidden state $h_t$ of the ConvGRU can be described by the following equations:

$$z_t = \sigma(W_{xz} * x_t + W_{hz} * h_{t-1} + b_z), \tag{8}$$

$$r_t = \sigma(W_{xr} * x_t + W_{hr} * h_{t-1} + b_r), \tag{9}$$

$$\tilde{h}_t = \tanh(W_x * x_t + W_h(r_t \circ h_{t-1}) + b_h), \tag{10}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t, \tag{11}$$

where $*$ denotes convolution, weight matrices $W_{xz}, W_{hz}, W_{xr}, W_{hr}, W_x, W_h$ are 2D-convolutional kernels, and $b_z, b_r, b_h$ are the corresponding bias terms. In order to preserve the same spatial size of hidden representations, zero-padding is used in recurrent convolutions. The ConvGRU architecture is illustrated in Figure 2.

Our main idea of using convGRU unit is similar to ((Maslov and Makarov, 2020)), in which authors used combinations of convolutions and recurrent blocks for temporal information aggregation for supervised depth estimation. We believe that in self-supervised setting temporal consistency is even more essential for robust depth estimation in both, single image and sequential settings. The key component to it is properly chosen self-attention mechanism described next.

## 3.4 Self-attention

Convolutional neural networks are widely used in many neural network architectures, especially in computer vision. However, a single convolutional layer can only capture dependencies in a small neighborhood. Therefore, alternative approaches based on self-attention are also starting to arise (Ramachandran et al., 2019). Self-attention can be considered as an independent layer that can be a good substitute for convolutions. Experiments by Ramachandran et al. (2019) show that simply replacing spatial convolutions with self-attention in a ResNet-type architecture leads to higher accuracy in ImageNet classification and COCO object detection. In this work, we use self-attention layers to fuse temporal features.

Next, we describe the self-attention layer used in our architecture in detail; our exposition directly follows the original paper by Ramachandran et al. (2019). Similar to a convolution, given a pixel $x_{ij} \in \mathbb{R}^{d_{in}}$, we first take a neighborhood $N_k(i, j)$ centered on this pixel that extends for $k$ pixels around $x_{ij}$. This region is called a memory block.

One-headed attention is calculated as follows:

$$y_{ij} = \sum_{a,b \in N_k(i,j)} \text{softmax}_{ab}(q_{ij}^T k_{ab}) v_{ab}, \tag{12}$$

where $y_{ij} \in \mathbb{R}^{d_{out}}$, queries $q_{ij} = W_Q x_{ij}$, keys $k_{ab} = W_K x_{ab}$, values $v_{ab} = W_V x_{ab}$. $W_Q, W_K, W_V \in \mathbb{R}^{d_{out} \times d_{in}}$ are linear transforms. Similar to convolutions, self-attention aggregates spatial information over a
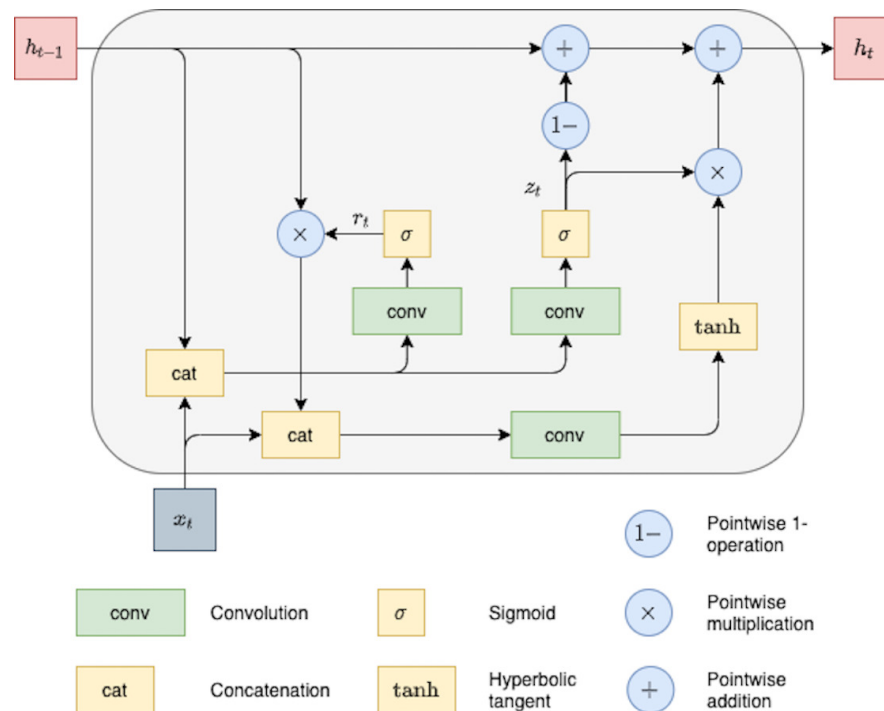
**Figure 2.** ConvGRU block with feature map as input.

neighborhood region but aggregation is a convex combination of value vectors with weights defined by the softmax operator. This procedure repeats for every pair of indices $(i, j)$. Moreover, it is possible to use multiple attention heads, splitting the pixels $x_{ij}$ depthwise into $N$ groups $x_{ij}^{(n)} \in \mathbb{R}^{d_{in}/N}$, computing the output for each group separately, and then concatenating these outputs.

Next, to aggregate information from different resolutions, frames, and components, we need efficient fusion model taking the best out of convolutional and attention aggregation mechanisms.

### 3.5 Fusion

Fusion is a technique commonly used to aggregate multimodal features. Multimodality implies that objects have different semantic characteristics. Traditional fusion methods include concatenation or simple averaging. Numerous studies employ fusion for processing multimodal inputs, and many more complex fusion methods have been proposed over the last decade.

In this work, to fuse temporal features we use residual convolutional units as introduced by Lin et al. (2016) and their modification that we call *residual self-attention units*. A residual convolution unit itself is a simplified version of the convolution unit in the original ResNet (He et al., 2015). It consists of two spatial convolutions with ReLU activations and batch normalization. It also has a skip connection between the input and the output of convolution layers. In this work, we propose a modification of the residual convolution unit: we replace the spatial convolutional layer with a self-attention layer. We believe that this modification helps capture dependencies from different neighborhoods of the input's representation, potentially resulting in more accurate predictions. Figure 3 shows a schematic overview of the residual self-attention unit (Fig. 3B) and residual convolutional unit (Fig. 3C).

Residual convolutional and self-attention units are used to fuse ConvGRU outputs and hidden states. Fusion modules progressively combine and upscale feature maps in order to generate a fine-grained prediction. Figure 3A presents a schematic overview of our Fusion block.

Finally, we are ready to describe overall model.

### 3.6 Recurrent self-supervised depth estimation

In this section, we describe the depth network pipeline and illustrate the role of recurrent blocks and fusion units in it. For this study, we have attempted to investigate and propose a modification to existing pipelines that would utilize temporal information across frame sequences. To examine the impact of
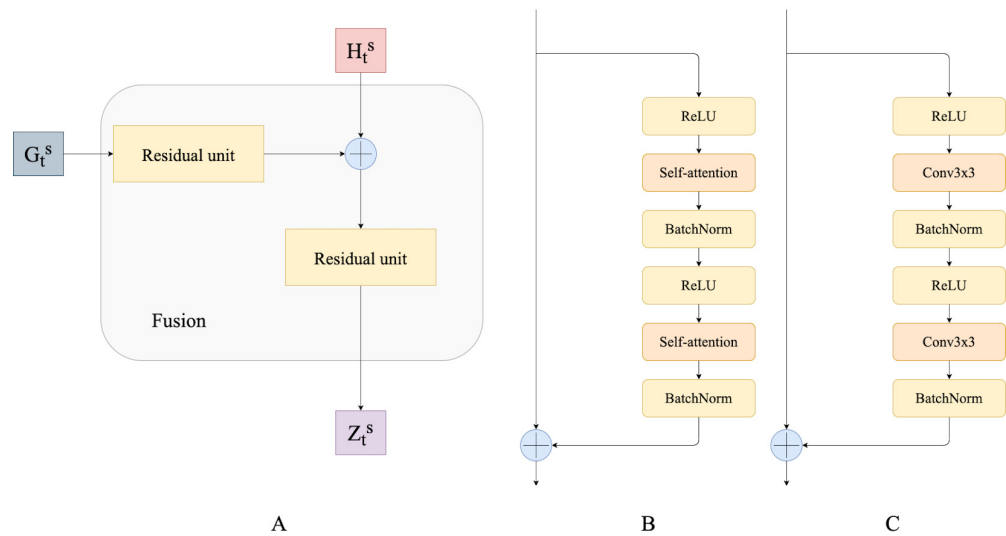
**Figure 3.** (A) fusion block; (B) residual self-attention unit; (C) residual convolutional unit.

temporal knowledge, we tested different modifications, and in this section, we describe them in more
detail. We have experimented with depth network components, while the pose network remains the same
as in previous models (Godard et al., 2019; Guizilini et al., 2020a; Zhou et al., 2017). The pose network
consists of the ResNet18 encoder and pose decoder that is presented in Figure 4.
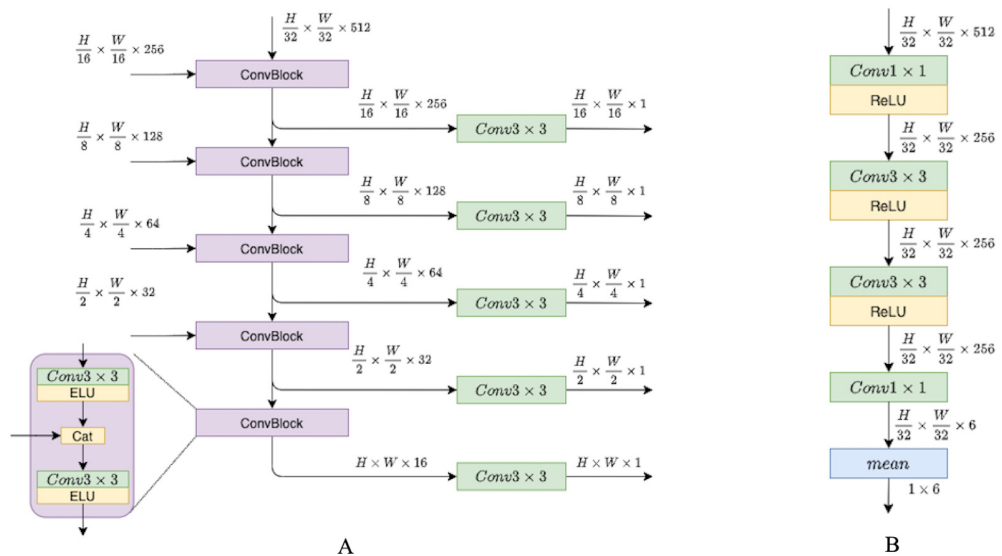
**Figure 4.** (A) Depth decoder. (B) Pose decoder. progressively upsample and fuse depth encoder outputs.
Pose encoder outputs pass through convolutions layers with ReLU activations, then flattened to six
degrees of freedom translation.

The key feature of the proposed depth model is the use of two types of temporal information; it is
schematically illustrated in Fig. 5. The first type of temporal information is the content of previous frames
transferred via a recurrent block, i.e., via the hidden state. The second type of temporal knowledge comes
from the dependencies between different scales of decoder outputs (indicated by green arrows in Fig. 5).
The architecture design is inspired by Sun et al. (2021) who used similar types of temporal knowledge
and connections between them for the real-time 3D scene reconstruction task. In (Sun et al., 2021), the
recurrent part of the model is based on 3D convolutions, and in this work we simplify their approach and
adapt it to our task.
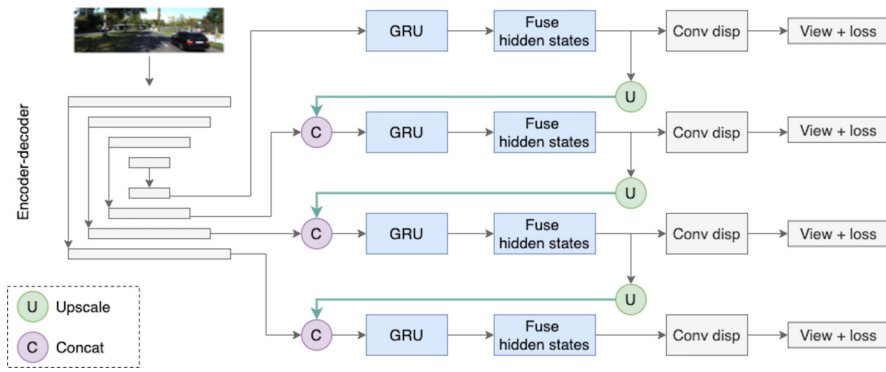
**Figure 5.** The Depth Net pipeline. First, a target frame is passed through the encoder-decoder architecture. The decoder produces outputs on four different scales, they are passed through recurrent ConvGRU and Fusion blocks, and finally, the depth prediction comes from upscaled maps passed through the convolution disparity layer. Road images and ground truth depth maps taken from (Menze and Geiger, 2015).

The depth model consists of three consecutive parts. The first part is an encoder-decoder block with skip connections, where the encoder is a ResNet18 and the decoder is a sequence of upsampling convolutional blocks as shown in Figure 4. The decoder produces outputs on four different scales. We consider two types of decoder outputs: single-channel outputs and multi-channel outputs with $n \in \{16, 32, 64, 128\}$ channels. In the decoder architecture, multi-channel outputs are usually mapped to single-channel ones via a convolutional layer, as in (Godard et al., 2019). However, in this work we do not complete the depth prediction process at this stage but rather continue to process internal feature representations produced in the decoder. These representations are fed into the second part of our model, recurrent ConvGRU blocks. ConvGRU on a particular scale takes as input a concatenation of the depth output at this scale and an upscaled representation from the previous scale (except for, obviously, the first scale where ConvGRU receives only the depth output). The third part of the model is the Fusion block that helps aggregate temporal information. The head of the model consists of simple convolution blocks that produce disparity for a target image.

Next, we give a formal description of the full depth estimation pipeline for a given image. Let $X_t$ be an input image, and let $(H_t^1, H_t^2, H_t^3, H_t^4)$ be the hidden states of the recurrent block on four scales produced from previous frames. The encoder-decoder network produces outputs on four scales $s \in \{1, 2, 3, 4\}$. We denote by $F_t^s$ the encoder-decoder output on scale $s$ at time $t$:

$$(F_t^1, F_t^2, F_t^3, F_t^4) = f_{Dec}(f_{Enc}(X_t)). \tag{13}$$

Then, we feed decoder outputs $F_t^s$ and upscaled maps from previous scales into ConvGRU blocks:

$$G_t^s = f_{ConvGRU}^s(concat(F_t^s, U_t^{s-1})), \quad s \in \{2, 3, 4\}, \tag{14}$$
$$G_t^s = f_{ConvGRU}^s(F_t^s), \quad s = 1. \tag{15}$$

Next, we fuse ConvGRU outputs and hidden states from the previous timestamp:

$$Z_t^s = f_{Fusion}^s(G_t^s, H_t^s). \tag{16}$$

Fusion outputs then go to an upscaling block, resulting in $U_t^s$, and to the convolution disparity block that produces final depth map on this scale:

$$U_t^s = f_{Upscale}^s(Z_t^s), \quad s \in \{0, 1, 2\}, \tag{17}$$
$$\hat{D}_t^s = f_{ConvDisp}^s(Z_t^s), \quad s \in \{0, 1, 2, 3\}. \tag{18}$$
$$\tag{19}$$

For upscaling, we use bilinear interpolation while working with single-channel decoder outputs and the PixelShuffle operation (Shi et al., 2016) while working with multi-channel decoder outputs.

**Table 1.** Ablation study for single-image based methods. Results for different variants of our model with monocular training on KITTI Eigen split.

| Model | Eval | Dec. output | Upscale & concat | Fuse: ResConv | Fuse: ResSA | RMSE | RMSE log | Abs Rel | Sq Rel | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
|-------|------|-------------|------------------|---------------|-------------|------|----------|---------|--------|-----------------|-------------------|-------------------|
| Baseline | image | 1-ch | - | - | - | 5.512 | 0.223 | 0.140 | 1.610 | 0.852 | 0.946 | 0.973 |
| Monodepth2 | image | 1-ch | - | - | - | 4.862 | **0.193** | **0.115** | 0.903 | **0.877** | **0.959** | 0.981 |
| Model #1 | image | 1-ch | - | - | - | 5.615 | 0.216 | 0.137 | 1.007 | 0.816 | 0.941 | 0.978 |
| Model #2 | image | 1-ch | + | - | + | 5.123 | 0.203 | 0.129 | **0.861** | 0.846 | 0.953 | **0.982** |
| Model #3 | image | 1-ch | + | + | - | 4.982 | 0.199 | 0.120 | 0.877 | 0.861 | 0.955 | 0.981 |
| Model #4 | image | n-ch | - | - | - | 4.998 | 0.201 | 0.123 | 0.891 | 0.857 | 0.956 | 0.981 |
| Model #5 | image | n-ch | + | + | - | **4.851** | 0.200 | 0.124 | 0.875 | 0.859 | 0.956 | 0.980 |

**Table 2.** Ablation study for sequence-based methods. Results for different variants of our model with monocular training on KITTI Eigen split.

| Model | Eval | Dec. output | Upscale & concat | Fuse: ResConv | Fuse: ResSA | RMSE | RMSE log | Abs Rel | Sq Rel | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
|-------|------|-------------|------------------|---------------|-------------|------|----------|---------|--------|-----------------|-------------------|-------------------|
| Model #1 | seq | 1-ch | - | - | - | 5.198 | 0.208 | 0.133 | 0.926 | 0.828 | 0.948 | **0.981** |
| Model #2 | seq | 1-ch | + | - | + | **4.811** | **0.199** | 0.126 | 0.910 | 0.858 | **0.957** | **0.981** |
| Model #3 | seq | 1-ch | + | + | - | 4.976 | **0.199** | **0.120** | **0.874** | **0.861** | 0.955 | **0.981** |
| Model #4 | seq | n-ch | - | - | - | 4.963 | **0.199** | 0.122 | 0.879 | 0.859 | 0.956 | **0.981** |
| Model #5 | seq | n-ch | + | + | - | 4.847 | **0.199** | 0.124 | 0.883 | 0.860 | **0.957** | **0.981** |

# 4 EXPERIMENTS

In this section, we first describe our experimental setup, starting with the dataset, then providing the training details and baseline architectures, and finally detailing the system configuration.

## 4.1 KITTI Dataset

In order to compare the results with previously proposed monocular self-supervised methods, we evaluate our models on the KITTI 2015 dataset (Menze and Geiger, 2015). The data consists of outdoor driving scenes which usually include up to 15 cars and 30 pedestrians. Accurate ground truth depth maps that are needed for further evaluation are also provided.

We use the data split proposed by Eigen and Fergus (2014), with monocular frame sequences. We use the preprocessing technique suggested by Zhou et al. (2017) that removes static frames. In order to train the model on sequences, we randomly sample consecutive frames from available scenes. As a result, the training set contains 2685 monocular samples, and the validation set contains 176, where a sample is a sequence of frames. The length of the sequence is set to 10. Following Godard et al. (2019), we use the same camera intrinsics for all images, setting the principal point of the camera to the center of the image and focal length to the average of all focal lengths in KITTI. Training on image sequences is a challenging task for GPU memory limits. Therefore, we had to resize all input images to resolution $194 \times 640$. The same resizing technique was used in previous works (Godard et al., 2019; Guizilini et al., 2020a; Zhou et al., 2017), so comparison with existing models appears to be fair.

## 4.2 Training details

Working with recurrent neural networks introduces certain complications in the training strategy. We need to choose how to initialize hidden states and how to train the network. The approach to initializing hidden states with zeros is commonly used in many sequence-to-sequence natural language processing tasks, including machine translation (Parmar and Devi, 2019). For monocular sequences, we usually use relatively short hidden states because of limited memory resources. Therefore, the impact of the hidden state becomes very important. We follow Maslov and Makarov (2020) and Patil et al. (2020) and use two training stages. On the first stage, we train a model for hidden states considering them as parameters, at the second stage we use learned hidden states as initial states for every sequence.

All models are trained for 20 epochs with batch size 2, where one sample in the batch is one sequence of length 10. We use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, learning rate $10^{-4}$ for the first 15 epochs which is then reduced to $10^{-5}$ for the remainder. The smoothness term $\lambda$ is set to 0.001, and

**Table 3.** Comparison with state-of-the-art image-based methods on KITTI Eigen split. Input image resolution is 640x192. The first column classifies methods as 's' (supervised) or 'u' (self-supervised/unsupervised). Best results in each group are marked as **bold**.

| Model | Setting | RMSE | RMSE log | Abs Rel | Sq Rel | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
|---|---|---|---|---|---|---|---|---|
| DORN (Fu et al., 2018) | s | **2.727** | **0.120** | **0.072** | **0.307** | **0.932** | **0.984** | **0.994** |
| Struct2depth (Casser et al., 2018) | u | 5.291 | 0.215 | 0.141 | 1.026 | 0.816 | 0.945 | 0.979 |
| Monodepth2 (Godard et al., 2019) | u | 4.862 | 0.193 | 0.115 | 0.903 | 0.877 | 0.959 | 0.981 |
| PackNet SfM (Guizilini et al., 2020a) | u | 4.601 | 0.189 | 0.111 | 0.785 | 0.878 | 0.960 | 0.982 |
| Guizilini et al. (2020b) | u | **4.381** | **0.178** | **0.102** | **0.698** | **0.896** | **0.964** | **0.984** |
| Ours | u | 4.851 | 0.200 | 0.124 | 0.875 | 0.859 | 0.956 | 0.980 |

**Table 4.** Comparison with state-of-the-art sequence-based methods on KITTI Eigen split. Input image resolution is $640 \times 192$. The "Setting" column classifies methods as "s-seq" (supervised sequential, i.e., using previous frame information) or "u-seq" (self-supervised sequential). Best results in each group are marked in **bold**.

| Model | Setting | RMSE | RMSE log | Abs Rel | Sq Rel | $\delta < 1.25$ | $\delta < 1.25^2$ | $\delta < 1.25^3$ |
|---|---|---|---|---|---|---|---|---|
| Maslov and Makarov (2020) | s-seq | **4.104** | **0.170** | **0.101** | **0.707** | **0.887** | **0.964** | **0.986** |
| Luo et al. (2020) | u-seq | 4.876 | 0.205 | 0.130 | 2.086 | 0.878 | 0.946 | 0.970 |
| CoMoDA (Kuznietsov et al., 2021) | u-seq | 4.594 | 0.183 | 0.103 | 0.862 | 0.899 | 0.961 | 0.981 |
| McCraith et al. (2020) | u-seq | 4.275 | 0.173 | **0.089** | 0.747 | 0.912 | 0.964 | 0.982 |
| ManyDepth (Watson et al., 2021) | u-seq | **4.261** | **0.170** | 0.090 | **0.713** | **0.914** | **0.966** | **0.983** |
| Ours | u-seq | 4.811 | 0.199 | 0.126 | 0.910 | 0.858 | 0.957 | 0.981 |

SSIM weight $\alpha$ is set to 0.85. Following Godard et al. (2019), we use pretrained ImageNet weights for the encoder. We also apply the following augmentations: horizontal flips with 50% chance; brightness, contrast, saturation, and hue jitter with ranges of $\pm 0.2$, $\pm 0.2$, $\pm 0.2$, and $\pm 0.1$ respectively.

For the baseline model, we follow Godard et al. (2019): we take the U-net model, where the encoder is a ResNet18 (He et al., 2015), and the decoder consists of upconvolutional filters, use the average reprojection term for the loss function, and do not apply scaling to decoder outputs before computing the loss.

### 4.3 System configuration

All experiments were conducted using a free distribution of Anaconda with Python 3.8. The models were implemented with the Pytorch library (Paszke et al., 2019). We used an RTX6000 GPU with 32 GB RAM. The operating system was Ubuntu 18.04.5 LTS.

## 5 RESULTS

In this section, we present the evaluation results for our model and show that it produces competitive results by comparing it with existing self-supervised models. For evaluation metrics, we use standard depth evaluation metrics from (Eigen et al., 2014; Eigen and Fergus, 2014) and the Eigen split setting from (Eigen et al., 2014). As has usually been done for KITTI Eigen split evaluation (Godard et al., 2019; Watson et al., 2021), we clip predicted depths at 80m, and only evaluate on depth maps with ground truth under 80m.

### 5.1 Ablation Study

In order to estimate the effect of each component that we use in our pipeline, we provide an ablation study of the proposed modifications. An ablation study implies changing various components of our model, and its results help to better understand how each component contributes to the overall performance in monocular training. We first compare models with single-image evaluation settings (Table 1). Then we evaluate our models on sequences. The results are illustrated in Table 2. We can see that the baseline, without any of our modifications and contributions from Godard et al. (2019), performs the worst.

There are five different modifications; all models have the ConvGRU block but they differ in the number of channels in decoder outputs, the use of the upscaling block, and the use of the Fusion block. In particular, models #1 and #4 have only ConvGRU layers, other models use the Fusion block and upscaling, and differ only in Fusion block type.

Since our model was trained on sequences, we hypothesize that if we make use of sequence information at test time, we can achieve higher depth prediction accuracy. As a result, we propose two evaluation
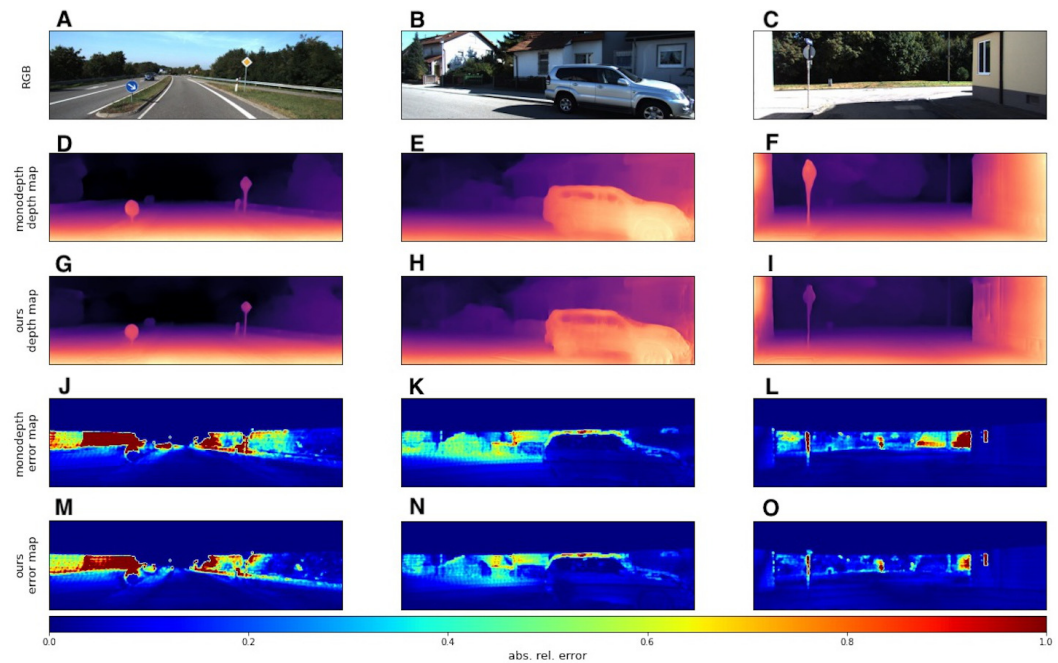
**Figure 6.** Qualitative comparison of our model with single-image evaluation setting and Monodepth2 (Godard et al., 2019). A, B, C correspond to RGB images. D, E, F correspond to depth output produced by *Monodepth2*. G, H, I correspond to depth output produced by our model. J, K, L correspond to error maps of *Monodepth2*. M, N, O correspond to error maps of our model. Brighter colors on error maps mean higher errors. Road images and ground truth depth maps taken from (Menze and Geiger, 2015).

settings: single-image evaluation and sequential evaluation. Single-image evaluation means that we process each test frame separately. The model, initialized with trained hidden states, takes a single test frame as input and produces a depth map. On the other hand, sequential evaluation implies that we run the model on the previous $n$ frames before producing the depth map for the target test image. We experimented with different values of $n$ and empirically found that 10 frames of history appear to be optimal. During sequential evaluation we "gather" temporal information from previous frames via hidden states and utilize this information in the final prediction for the test image. This sequential evaluation approach does reflect real-life scenarios well since in these scenarios the depth estimation component usually does have access to the frame history. As for the KITTI dataset, previous frames are available for all test images except for those at the beginning of a scene; for these frames, we use the maximum available number of previous frames during evaluation.

Despite the fact that most of our models still lose in the final quality in terms of some evaluation metrics to *Monodepth2*, we can make some interesting conclusions regarding the proposed components. For example, all models perform better when they use sequences during evaluation. The upscaling technique also gives a boost to the depth prediction quality. As for the Fusion block, residual self-attention units slightly improve the quality compared to residual convolutional units, as we expected. In contrast, it is hard to say that the use of multi-channel decoder outputs leads to significantly better results. Therefore, this component may be considered redundant, i.e., not contributing to the overall model performance.

In order to choose the best model, we focus on the RMSE metric. Since the errors are squared before averaging, the RMSE score gives a relatively high weight to large errors. Therefore, we consider RMSE to be more reasonable in the depth estimation task, where large errors are particularly undesirable. According to these considerations, we choose model #2 to be the best in our experiments.

Finally, we also provide results of speed tests on RTX6000 GPU. We find that our model runs in real-time with $124 \pm 16$ FPS on resolution $192 \times 640$ compared to DORN with $15 \pm 3$ FPS.
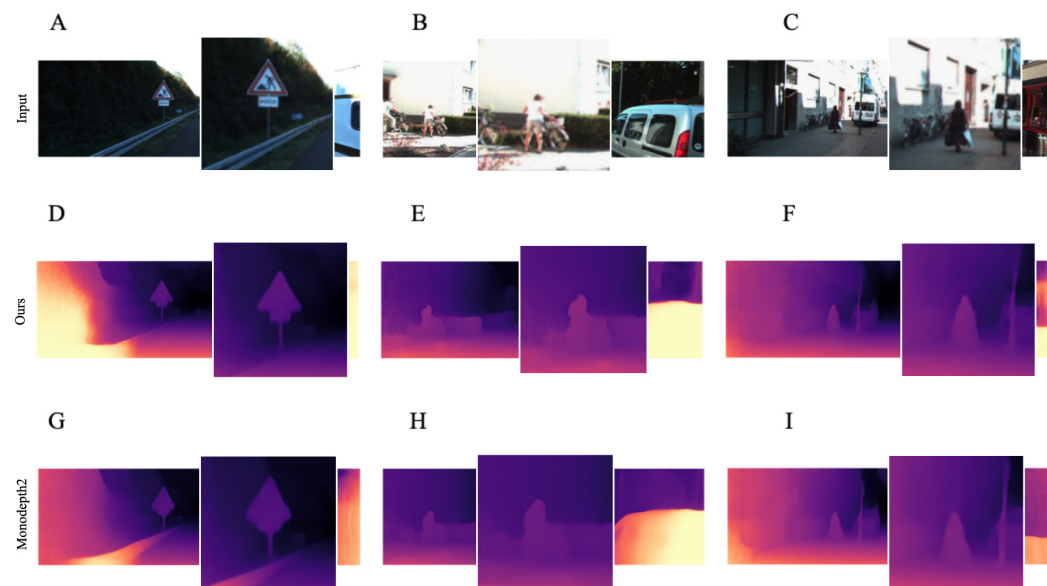
**Figure 7.** Examples where our model with single-image evaluation setting produces more accurate maps than *Monodepth2* (Godard et al., 2019) in the neighborhood of objects of non-trivial and angular shapes. A, B, C correspond to RGB images and their zoomed in regions. D, E, F correspond to the depth output produced by our model. G, H, I correspond to the depth output produced by *Monodepth2*. Road images and ground truth depth maps taken from (Menze and Geiger, 2015).

## 5.2 Results on the KITTI dataset

We compare our best model with previous single-image approaches such as (Godard et al., 2019; Guizilini et al., 2020a; Casser et al., 2018) and multi-frame sequential approaches such as (Kuznietsov et al., 2021; McCraith et al., 2020; Watson et al., 2021). The results are presented in Table 3 and Table 4, respectively. We provide scores on the KITTI Eigen split with input resolution $192 \times 640$. In addition, we compare to state-of-the-art supervised models such as (Fu et al., 2018; Maslov and Makarov, 2020). We observe that our approach outperforms most single-image self-supervised models in terms of the RMSE score. Our model achieves worse metrics than some sequential approaches, but next, we show that the proposed method better in qualitative comparison.

## 6 DISCUSSION

Our experiments have shown that the model benefits from information collected from previous frames. We assume that incorporation of the recurrent block into a self-supervised training pipeline may provide a potential boost in the results not only in our setting but also in supervised or online supervised training strategies.

We used the KITTI dataset to evaluate our models with standard metrics from Eigen et al. (2014); Eigen and Fergus (2014). In addition to quantitative results, we also provide several visual samples. We present a qualitative comparison with the *Monodepth2* model (Godard et al., 2019), since this model served as the basis for our modifications. In Figure 6 we provide some prediction examples and error maps that are calculated as the absolute difference between the ground truth and predicted depth maps. These visual results show that our model appears to be more accurate in far areas with higher depth variance. We also provide a qualitative comparison with ManyDepth (Watson et al., 2021) model, that is the current state-of-the-art self-supervised sequential method. The examples in Figure 8 illustrate that our model better predicts depth for moving objects like cyclists and cars.
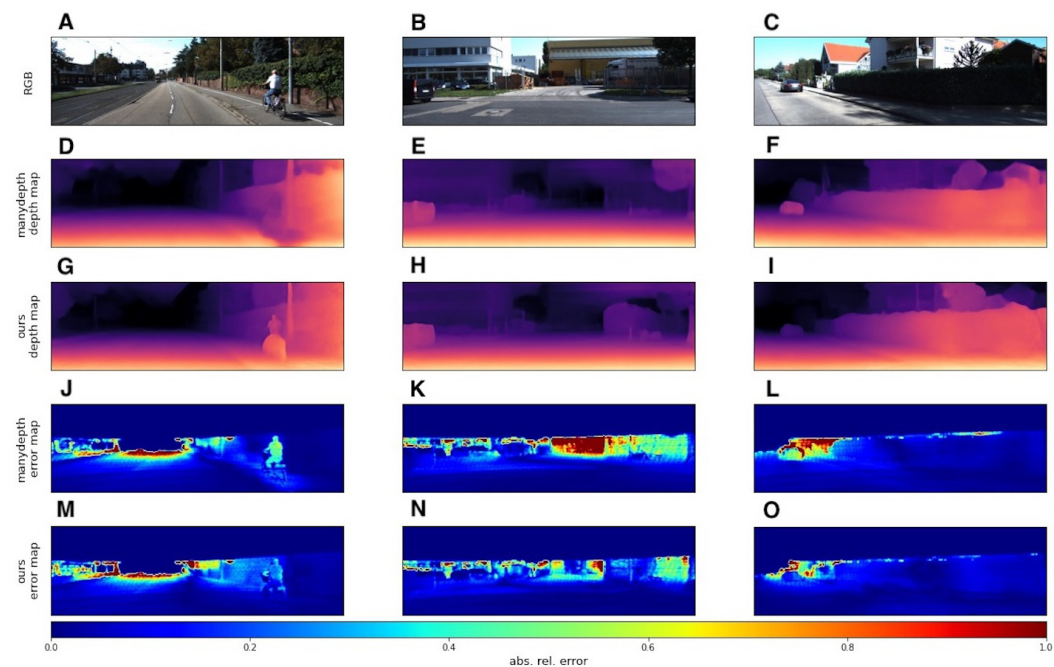
**Figure 8.** Qualitative comparison of our model with sequential evaluation setting and ManyDepth (Watson et al., 2021). A, B, C correspond to RGB images. D, E, F correspond to depth output produced by *ManyDepth*. G, H, I correspond to depth output produced by our model. J, K, L correspond to error maps of *ManyDepth*. M, N, O correspond to error maps of our model. Brighter colors on error maps mean higher errors. Road images and ground truth depth maps taken from (Menze and Geiger, 2015).

In addition, we noticed that our model produces more accurate depth maps in the neighborhood of objects with non-trivial and triangular shapes, such as people and road signs. We assume that the model performs quite reasonably in these regions because it captures historical information about the frames, resulting in more precise predictions. Qualitative proofs of this evidence are presented in Figure 7, where we zoom in on the most interesting regions.

Although our model does not achieve state-of-the-art performance in the self-supervised setting, it appears to outperform *Monodepth2* (Godard et al., 2019) and *ManyDepth* (Watson et al., 2021) in view synthesis quality. We measured the Frechet Inception Distance (FID) (Heusel et al., 2017) of reconstructed test images and saw that our method with FID of 16.863 visually outperforms Monodepth2 and ManyDepth with FID of 16.911 and 16.902 respectively. From the nature of FID, it means that the distribution of features from our model lies more closely to the distribution of features of the ground truth depth maps compared to MonoDepth2 and ManyDepth, which also leads to more precise boundary localization as shown in the qualitative comparison.

One important open question remains regarding the length of the input sequence. As we have mentioned before, we selected the optimal number empirically. However, we experimented only with short sequences of length up to 12 frames due to memory limits. Patil et al. (2020) also tested different sequence lengths and decided that 30 is the optimal value for their setting. In contrast, Maslov and Makarov (2020) used sequences consisting of 10 frames. We assume that this number might be different for different models, and it would be interesting to investigate the effect of the history length and devise optimization strategies to find the optimal value for this parameter for a given model and setting.

## 7 CONCLUSION

In this study, we have presented a self-supervised method that predicts depth from a single image or a sequence of images when available. We provided an ablation study testing different configurations of model components. Experiments show that we have achieved improvements over the basic architecture from both recurrent ConvGRU layer and Fusion module with a self-attention mechanism. We also

demonstrate that our method can be applied to real-time. It means that the use of previous frames during training and on test time can provide a significant boost in the accuracy of self-supervised depth estimation.

**Future work.** As an interesting direction of future work we suggest the following problems. First, it would be interesting to find an optimal sequence length in both training and evaluation modes. Second, we need to find stabilization techniques for training recurrent networks in a self-supervised pipeline. Third, we expect that our results could be further improved via more advanced modifications such as cost volume (Watson et al., 2021) or more stable feature-based losses (Shu et al., 2020).

## ACKNOWLEDGMENTS

The authors are grateful to Aleksei Karpov for his invaluable contribution to polishing the article presentation, qualitative comparison, and text.

## REFERENCES

Ballas, N., Yao, L., Pas, C., and Courville, A. (2015). Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*.

Cao, Y., Wu, Z., and Shen, C. (2016). Estimating depth from monocular images as classification using deep fully convolutional residual networks. *CoRR*, abs/1605.02305.

Casser, V., Pirk, S., Mahjourian, R., and Angelova, A. (2018). Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *CoRR*, abs/1811.06152.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Eigen, D. and Fergus, R. (2014). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *CoRR*, abs/1411.4734.

Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *CoRR*, abs/1406.2283.

Fu, H., Gong, M., Wang, C., Batmanghelich, K., and Tao, D. (2018). Deep ordinal regression network for monocular depth estimation. *CoRR*, abs/1806.02446.

Garg, R., G, V. K. B., and Reid, I. D. (2016). Unsupervised CNN for single view depth estimation: Geometry to the rescue. *CoRR*, abs/1603.04992.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Godard, C., Aodha, O. M., and Brostow, G. J. (2016). Unsupervised monocular depth estimation with left-right consistency. *CoRR*, abs/1609.03677.

Godard, C., Mac Aodha, O., Firman, M., and Brostow, G. J. (2019). Digging into self-supervised monocular depth prediction. In *The International Conference on Computer Vision (ICCV)*.

Graves, A., Mohamed, A., and Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778.

Guizilini, V., Ambrus, R., Pillai, S., Raventos, A., and Gaidon, A. (2020a). 3d packing for self-supervised monocular depth estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Guizilini, V., Hou, R., Li, J., Ambrus, R., and Gaidon, A. (2020b). Semantically-guided representation learning for self-supervised monocular depth. *CoRR*, abs/2002.12319.

Gupta, S., Girshick, R. B., Arbelaez, P., and Malik, J. (2014). Learning rich features from RGB-D images for object detection and segmentation. *CoRR*, abs/1407.5736.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

He, Y. (2017). Estimated depth map helps image classification. *arXiv preprint arXiv:1709.07077*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780. Based on TR FKI-207-95, TUM (1995).

Hu, J., Ozay, M., Zhang, Y., and Okatani, T. (2019). Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1043–1051. IEEE.

Hur, J. and Roth, S. (2021). Self-supervised multi-frame monocular scene flow. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Johnson, J., Alahi, A., and Li, F. (2016). Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155.

Kuznietsov, Y., Proesmans, M., and Van Gool, L. (2021). Comoda: Continuous monocular depth adaptation using past experiences. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2907–2917.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. *CoRR*, abs/1606.00373.

Lin, G., Milan, A., Shen, C., and Reid, I. D. (2016). Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *CoRR*, abs/1611.06612.

Luo, C., Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R., and Yuille, A. L. (2018). Every pixel counts ++: Joint learning of geometry and motion with 3d holistic understanding. *CoRR*, abs/1810.06125.

Luo, X., Huang, J., Szeliski, R., Matzen, K., and Kopf, J. (2020). Consistent video depth estimation. *ACM Transactions on Graphics*, 39(4).

Ma, F., Cavalheiro, G. V., and Karaman, S. (2018). Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. *CoRR*, abs/1807.00275.

Ma, F. and Karaman, S. (2017). Sparse-to-dense: Depth prediction from sparse depth samples and a single image. *CoRR*, abs/1709.07492.

Mahjourian, R., Wicke, M., and Angelova, A. (2018). Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. *CoRR*, abs/1802.05522.

Makarov, I., Aliev, V., and Gerasimova, O. (2017a). Semi-dense depth interpolation using deep convolutional neural networks. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1407–1415, NY, USA. ACM, ACM.

Makarov, I., Aliev, V., Gerasimova, O., and Polyakov, P. (2017b). Depth map interpolation using perceptual loss. In *Mixed and Augmented Reality (ISMAR-Adjunct), 2017 IEEE International Symposium on*, pages 93–94, NY, USA. IEEE, IEEE.

Makarov, I. and Korinevskaya, A. (2019). Fast depth map super-resolutionusing deep neural network. In *Mixed and Augmented Reality (ISMAR-Adjunct), 2018 IEEE International Symposium on*, pages 1–6, NY, USA. IEEE, IEEE.

Makarov, I., Korinevskaya, A., and Aliev, V. (2018a). Fast semi-dense depth map estimation. In *Proceedings of the 2018 ACM Workshop on Multimedia for Real Estate Tech*, pages 18–21, NY, USA. ACM, ACM.

Makarov, I., Korinevskaya, A., and Aliev, V. (2018b). Sparse depth map interpolation using deep convolutional neural networks. In *2018 41st IC on Telecommunications and Signal Processing (TSP)*, pages 1–5, NY, USA. IEEE, IEEE.

Makarov, I., Korinevskaya, A., and Aliev, V. (2018c). Super-resolution of interpolated downsampled semi-dense depth map. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology*, page 27, NY, USA. ACM, ACM.

Makarov, I., Maslov, D., Gerasimova, O., Aliev, V., Korinevskaya, A., Sharma, U., and Wang, H. (2019). On reproducing semi-dense depth map reconstruction using deep convolutional neural networks with perceptual loss. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1080–1084.

Maslov, D. and Makarov, I. (2020). Online supervised attention-based recurrent depth estimation from monocular video. *PeerJ Computer Science*, 6:e317.

Maslov, D. and Makarov, I. (2021). Fast depth reconstruction using deep convolutional neural networks. In *International Work-Conference on Artificial Neural Networks*, pages 456–467. Springer.

McCraith, R., Neumann, L., Zisserman, A., and Vedaldi, A. (2020). Monocular depth estimation with self-supervised instance adaptation. *CoRR*, abs/2004.05821.

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

Menze, M., Heipke, C., and Geiger, A. (2015). Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*.

Menze, M., Heipke, C., and Geiger, A. (2018). Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*.

Nikolenko, S. I. (2021). *Synthetic Data for Deep Learning*. Springer.

Parmar, M. and Devi, V. S. (2019). Neural machine translation with recurrent highway networks. *CoRR*, abs/1905.01996.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Paszke et al., A. (2019). Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Patil, V., Gansbeke, W. V., Dai, D., and Gool, L. V. (2020). Don't forget the past: Recurrent depth estimation from monocular video. *CoRR*, abs/2001.02613.

Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. (2019). Stand-alone self-attention in vision models. *CoRR*, abs/1906.05909.

Schönberger, J. L. and Frahm, J. (2016). Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shen, T., Zhou, L., Luo, Z., Yao, Y., Li, S., Zhang, J., Fang, T., and Quan, L. (2019). Self-supervised learning of depth and motion under photometric inconsistency. *CoRR*, abs/1909.09115.

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *CoRR*, abs/1609.05158.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214.

Shin, D., Ren, Z., Sudderth, E. B., and Fowlkes, C. C. (2019). Multi-layer depth and epipolar feature transformers for 3d scene reconstruction. *CoRR*, abs/1902.06729.

Shu, C., Yu, K., Duan, Z., and Yang, K. (2020). Feature-metric loss for self-supervised learning of depth and egomotion. *CoRR*, abs/2007.10603.

Siam, M., Valipour, S., Jägersand, M., and Ray, N. (2016). Convolutional gated recurrent networks for video segmentation. *CoRR*, abs/1611.05435.

Sun, J., Xie, Y., Chen, L., Zhou, X., and Bao, H. (2021). Neuralrecon: Real-time coherent 3d reconstruction from monocular video. *arXiv preprint arXiv:2104.00681*.

Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., and Fragkiadaki, K. (2017). Sfm-net: Learning of structure and motion from video. *CoRR*, abs/1704.07804.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612.

Watson, J., Aodha, O., Prisacariu, V., Brostow, G., and Firman, M. (2021). The temporal opportunist: Self-supervised multi-frame monocular depth. *arXiv preprint arXiv:2104.14540*.

Xie, J., Girshick, R. B., and Farhadi, A. (2016). Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. *CoRR*, abs/1604.03650.

Yang, N., Wang, R., Stückler, J., and Cremers, D. (2018). Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. *CoRR*, abs/1807.02570.

Yin, W., Liu, Y., Shen, C., and Yan, Y. (2019). Enforcing geometric constraints of virtual normal for depth prediction. *CoRR*, abs/1907.12209.

Zhang, H., Shen, C., Li, Y., Cao, Y., Liu, Y., and Yan, Y. (2019). Exploiting temporal consistency for real-time video depth estimation. *CoRR*, abs/1908.03706.

Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. *CoRR*, abs/1704.07813.