Influence of Biological Cues on Monocular Depth Estimation

Jochem Groot Roessink
University of Twente
The Netherlands
j.grootroessink@student.utwente.nl

Abstract

Monocular depth estimation (MDE) in computer vision is the process of estimating the distance to the camera for every pixel in a single 2D image, with that image being the only input data. In theory, this task is problematic since an infinite amount of 3D scenes can generate the same 2D image. Fortunately, for most real-world images there is little ambiguity for what its 3D scene should look like. In fact, current MDE methods exist that estimate depth that is close to the corresponding measured depth. Humans are also able to estimate depth from single-eye observations using prior knowledge about how certain cues in their observation (e.g. the size of a familiar object) relate to depth. In this work, these biological depth cues are extracted from an image and encoded into extra image channels. This new extended image is used as the input for an MDE method, which allows for studying the effects of explicit biological cues on M-DE. Observed effects in this study are substantial increases in estimation performance and efficiency in terms of training data. Furthermore, while the positive effects of cues on performance are most apparent for small amounts of data, the effects still remain substantial for much larger amounts of data. These results can be attributed to the fact that cue methods provide prior knowledge to an MDE method that it cannot learn from its training set. Since the cue method that contributes to the greatest performance increase uses unsupervised training, it provides a way of improving MDE using only unlabelled images. As a further contribution, a data set containing over 100K image-depth pairs is created using a realistic virtual environment.

Keywords: Monocular Depth Estimation, Biological Depth Cues, Prior Knowledge

1. Introduction

Depth estimation (DE) in computer vision is the process of estimating the distances to observed objects from (stereo) image or video data. Monocular depth estimation (MDE) is generally accepted as achieving this using only a single image, i.e. estimating the distance to the camera for every pixel in the image. There are many areas where MDE is used, these include autonomous driving [1], augmented reality [48] and 3D reconstruction [56].

A specific case of 3D reconstruction is image/video anonymisation, which was the topic of the thesis of Conde Moreno [12]. Since the used MDE method was found to be the bottleneck, the successive work from Al-jibouri created a replacement for the MDE method that uses a conditional Generative Adversarial Network (cGAN) [2]. This work directly follows from that of Al-jibouri, with the goal of further improving the MDE while studying the effects of explicit biological depth cues (hereinafter also referred to as just 'cues'). These cues refer to the different phenomena that humans or other animals use to perceive the depth in the environment observed by their vision. The cues included in this work are **object size**, **linear perspective**, **foreshortening**, **overlap**, **blur** and **texture gradient**, which are all described in Subsection 1.2.

The idea behind using cues is that they provide prior knowledge that can positively influence an MDE method. Such knowledge could be relevant for the depth of an image, while the MDE method cannot learn it from the image data, either due to limited capacity or the information simply not being present in its training set. Auty and Mikolajczyk have shown that the addition of explicit object size information can improve the performance of an MDE method [3]. This study provides methods to extract information related to one or more of these cues from a single image a encode this into extra image channels. Then, instead of a 3-channel (red-green-blue or RGB) image, an N-channel (original image with cue channels) image is used as the input for an MDE method.

A thorough search of the relevant literature has not yielded any work where the same approach was applied with a different method for object size or with other cues. With that in mind, this work provides a novel method for extracting object size information that includes foreshortening information and uses unsupervised training. Furthermore, methods for extracting linear perspective, overlap, blur and

texture gradient are all novelly applied using this approach. Additionally, a more explicit study in terms of data efficiency is applied, while the effects of cues on training efficiency (rate of convergence), and stability of training are studied as well. Finally, how the influence of cues on an MDE method changes when more training data becomes available is studied.

The remainder of this section consists of a listing of research questions and requirements, as well as a description of the incorporated cues. Section 2 provides a review of the literature relevant to this study. A custom virtual data set that is used in some of the experiments in described in Section 3. In Section 4, the model that produces a depth map from images that contain cue channels is described, as well as the methodology for extracting these cue channels from a single image. It also describes the experiments that are performed to answer the research questions, including the data set and evaluation metrics these experiments require. Section 5 describes the results of the experiments, while Section 6 provides a discussion on them. The conclusion of this study, including answers to the research questions and a description of future work is described in Section 7.

1.1. Research Questions and Requirements

Research questions The introduction above describes the research that is to be done in this work, which leads to the following research questions:

- **RQ1**: How does the inclusion of biological cues affect an MDE method?
- **RQ2**: How does an MDE method that uses biological cues compare to the same method without cues but with access to more training data?
- **RQ3**: How does an increase in the amount of training data affect the influence of biological cues?
- **RQ4**: How does the inclusion of biological cues affect state-of-the-art MDE methods?

Requirements The definition of MDE leads to the following requirements for the input of an MDE method or any cue extraction method:

- **Context-free**: the method should function for a single image that is not from a sequence or a binocular pair.
- **Offline**: the method should function for an image that is already captured and is not able to control camera movement or lens accommodation.

1.2. Biological Depth Cues

The MDE requirements limit which biological depth cues can be used. The cues incorporated for this work are the known cues that are in line with these requirements and operate at a relevant depth scale. Figure 1 provides visual examples.

Linear Perspective | Blur & Foreshortening | Overlap | Texture Gradient

Figure 1. Visual examples of cues.

Object size When a person recognises an object of which they know the approximate dimensions, this knowledge provides insight into how far away it is. Additionally, when multiple similar objects are observed, their difference in observed size is directly related to their relative distance, and consequently, their depth. Since different extraction methods are used for these two cases of object size, they are referred to as *familiar size* and *relative size*, as coined by Auty and Mikolajczyk [3].

Linear perspective Two lines that are parallel in a 3D environment converge to a single vanishing point in 2D observations (unless the line of sight of the observer is perpendicular to any third parallel line). Consider a point on a texture that is enclosed by two of these originally parallel lines (for example, a road enclosed by sidewalks). Then, the distances to both lines in such an observation are related to the depth of that point. The further away a point is from the observer, the shorter the summed distance to both line is [8, 41].

Foreshortening This convergence present in linear perspective applies to objects without straight edges as well. An example of this would be a person that stretches their arm out along the line of sight of an observer, with the hand being near the observer. In this case, the hand is observed to be significantly larger, relative to the rest of the body. Among artists, the technique of replicating this effect is called foreshortening. Ivanov et al. has shown that foreshortening can be used to determine the degree to which an object converges [25].

Overlap If two opaque objects overlap in an observation, then the one that is further away from the observer is partly obscured by the other. This feature provides a person with insight into which of the two is positioned at a closer distance.

Blur Every convex lens has a focal point: the point at which originally parallel light rays meet after being refracted by the lens. An eye observes an object positioned at this point as sharper than its surroundings. The further away an object is from the focal point, the blurrier the observation of it will be. If two distinct regions in the image border each other, then the sharper one is closer to the focal point. This is not yet enough information to determine which of the regions is closer to the observer. If the border between the two regions is sharp, the sharper object is expected to be closer to the observer. If this border is blurry, the blurrier object is likely closer. To summarize, region blur informs about the distance of a region to the focal point, while border blur aids in disambiguating whether a region is closer or farther away from the focal point. This provides direct insight into the relative distance of observed objects [37].

Texture gradient For a uniform texture (for example a brick road or a field of grass), the parts closer to the observer generally are perceived to be more detailed. For example, while individual blades of grass could be identified close to an observer, they unify into almost solid green if they are further away. Therefore, the variation in the coarseness of uniform textures can be used as a depth cue [41].

Other biological cues include **binocular disparity**, **accommodation**, **motion parallax**, **light scattering** and **shading**. However, the inclusion of the former three would require a method that is not in accordance with the requirements, while extraction of the latter two would only provide insight into depth at scales not relevant to this work.

2. Related Work

This section is a review of the relevant literature, structured in the following segments: disparity, end-to-end, cGAN, classification-regression, biologically inspired, and biology in DE. Additionally, a short conclusion is provided.

Disparity Estimating depth using binocular image pairs can be treated as a matching task: a pixel in one image is matched to a pixel in the other image. Each matching can be used to determine a disparity value for each pixel. Determining depth from disparity is a trivial process if the camera configuration is known [42, 46]. Žbontar and LeCun provided a method that uses a convolutional neural network (CNN) to match the pixels of two images [61].

Unlike methods that explicitly match pixels, end-to-end methods take a binocular image pair as input and produce a disparity map directly. Mayer et al. modified the optical flow estimator FlowNet [13] into the disparity estimator DispNet [38] to achieve this. The method from Godard et al. is also trained on binocular pairs to learn to produce disparity. Unlike DispNet, it takes only one image as input. From this image, it estimates disparity and reproduces the other image, which is compared to the original other image. Because of this, it can be used to predict disparity (and depth) from a single image after training [17].

While disparity methods allow for depth estimation, the required training data is not in line with the requirements from Subsection 1.1.

End-to-end Instead of estimating the disparity from an image (pair), there are methods that directly generate depth from a single image. While methods that do not use neural networks for this have been released in the past [47], more recent methods do use neural networks. These networks take an RGB image as input and estimate a depth map of the same resolution. This is often done using an encoder-decoder structure, where the image is encoded into latent space representation, which is decoded into a depth map. This estimation is compared to the actual depth values to determine the loss that is used during training.

Eigen et al. provided such a method, particularly one that uses two CNNs. The first CNN generates a complete depth map for a low-resolution version of the input image, while the second CNN takes both this depth map and a smaller patch of the input image as input. The details in this patch should improve upon the original low-resolution depth map [14]. The method from Laina et al. was introduced later and achieves MDE using a single CNN. This method has noticeably better performance [31].

A more recent method is BTS [32], which also uses a CNN architecture. Unique to this method is its decoder. At every stage of decoding the latent space representation a map of the same size as the input image is created using so-called local planar guidance. Together these maps are used to generate the final depth map. This method was introduced relatively recently and generates substantially better performance than Laina et al..

cGAN A cGAN consists of two networks: a generator and a discriminator. The generator works similarly to the previous methods, the discriminator is trained to distinguish between real depth maps and those produced by the generator. The classifications of the discriminator function as the supervisory signal to the generator, so that the latter learns to fool the discriminator.

Zhang et al. [57] and Chen et al. [10] both provided such a method, and both outperform the method from Laina et al. in every metric used. Similarly, for their thesis, Al-jibouri used a cGAN to estimate depth. This thesis is also the work from which this study directly follows. While it was not able to outperform Eigen et al. in the original work, the

method is re-implemented to be the baseline for this work, where it is able to substantially outperform Eigen et al..

Classification-regression While the previous methods are regression methods, depth estimation can also be treated as a classification problem. In this case, the classes refer to a subdivision of the total depth range, known as 'bins'. For each pixel of an image and for each bin, the probabilities that the depth value of the pixel lies within the range of that bin are produced. The supervisory signal is based on what bin the ground truth depth value lies in. Classification-regression uses this method to produce a continuous depth map. The depth value for each pixel is computed by linear combination of the probabilities with the centre value of their respective bin.

Bhat et al. provided such a classification-regression method, named AdaBins (adaptive bins). As the name suggests, the method learns to optimise the subdivision of the depth range into bins for each image, instead of using a fixed subdivision. The reason for this is that since different images contain different depth ranges, they benefit from different depth subdivisions. The method uses the encoder-decoder backbone EfficientNet-B5 [53] to produce a feature map for each image. From this feature map, both the bin ranges (and thus the centres) and the probabilities for each pixel for each bin are determined. Linear combination of the centres and probabilities produces the final depth map. This method greatly outperforms the method from Laina et al. for every metric used [5].

Li et al. created a similar method to AdaBins, named BinsFormer. The method uses the Swin transformer backbone [35] and a decoder structure to produce pixel representations. A separate transformer decoder is used to produce a length and an embedding for each bin. The lengths are used to create the division of the depth range into bins, while the embeddings are combined with the pixel representations to produce the probability maps. Again, the bin centres and probabilities are used to generate the final depth map. BinsFormer outperforms AdaBins [33].

Xie et al. investigated the effects of masked image modeling (MIM) on several computer vision tasks. For MDE, they pre-trained the SwinV2-L backbone of BinsFormer using MIM. This approach had a positive effect on the performance. At the time of writing, the resulting model has achieved the greatest performance on the KITTI Eigen split [16, 14], a widely used benchmark for MDE [55].

Biologically inspired Most previous biologically inspired computer vision work is focused on replicating the behaviour of certain cells, with the goal of achieving similar functionality to the observed functionality of those cells. This idea is similar to what led to the invention of the perceptron [39], which was inspired by the behaviour of a single neuron and formed the basis for neural networks.

Similarly, it has been suggested that simple cells in the visual cortex can be modelled by 2D Gabor filters [15]. Furthermore, convolutional kernels learned by the first convolutional layer of a CNN trained on pictures seem to resemble such Gabor filters [28]. This would explain the presence of CNNs in computer vision tasks, the popularity of which is emphasised by the fact that much of the previously described work makes use of CNNs.

However, regular CNNs still have limitations, an example of which is their robustness against unseen noise. Strisciuglio et al. created a new layer for CNNs, which is inspired by the push-pull inhibition phenomenon, which is observed in certain neurons in the visual cortex. Replacing the first layer of a CNN with this push-pull layer, significantly enhanced its robustness against unseen noise [52].

Not all biologically inspired computer vision work is about replicating phenomena observed in neurons in the visual cortex, Chen et al. discussed the use of an event-based neuromorphic vision sensor for autonomous driving. In contrast to regular cameras, this sensor does not record and process the brightness for every pixel for every frame, but observes brightness changes and triggers asynchronous events based on that, which is more similar to how human eyesight works. The advantages of this over regular cameras include low energy consumption, high dynamic range and no issues related to motion blur [9].

Biology in DE In regards to depth estimation, Mansour et al. compared the performance of two biological depth cues, binocular disparity and motion parallax. To achieve this, they used a moving robot fitted with binocular cameras for input and a LiDAR scanner as the ground truth. For binocular disparity, the pixel disparity between both cameras is converted to depth using triangulation. For motion parallax, the depth is determined in the same way but instead of using two cameras two different frames of a single camera are used (given that movement has occurred and that the movement is tracked). The results show that binocular disparity performs the best for shorter distances, while motion parallax performs better at longer distances. Additionally, it was found that combining both predictions generated the best results [36]. While binocular disparity and motion parallax are not in line with the requirements of MDE, the work does provide useful insight into the relative and combined performance of biological depth cues.

More specific to MDE, Auty and Mikolajczyk applied biologically inspired methods to AdaBins [5]. Instead of using just a three-channel (RGB) image, their work explicitly extracts information related to the object size cue and encodes this into extra image channels. What results is an image with 73 channels that functions as the input for

AdaBins. Since a scaled-down version of the EfficientNet encoder-decoder was used (B1 instead of B5), the method was not able to outperform AdaBins-B5. However, it was able to significantly improve the performance of the baseline AdaBins-B1 [3]. Therefore, this work has shown that the addition of explicit biological cue information to the input space can improve the performance of an MDE system.

Conclusion This literature review describes and compares different methods for MDE, each having different advantages and disadvantages. Therefore, the described studies provide useful insights for this work. BTS [32], AdaBins [5] and BinsFormer [33] are of particular interest for studying the state-of-the-art in this work. Additionally, the biologically inspired work shows how taking inspiration from biological vision systems can have a positive impact on the development of computer vision methods. Most relevant to this work is the work of Auty and Mikolajczyk, which shows that the explicit addition of object size information can improve the performance of an MDE system [3]. What has not been addressed in the literature above, nor has been found in any of the other relevant literature, is a study into the effects on MDE in the same manner with a different method for object size. Additionally, no study has been found where a method for linear perspective, blur, texture gradient, foreshortening or overlap is applied in the same way. Furthermore, no study into the effects of these cues in terms of training efficiency, stability during training and data efficiency has been found. Finally, a study into how the effects of cues change for increasing amounts of available data has also not been found. Therefore, what will be addressed in this work (as mentioned in Section 1) will bring novelty.

3. Contributed Depth Data

For this work, a custom data set is created using the video game engine Unreal Engine 5[1].

Motivation To answer the data efficiency questions **RQ2** and **RQ3**, different subsets of a data set are used, with the smallest subset using just 10% of the data (further described in Section 4.3). When this set is too small, the results of an experiment using it could be overly dependent on certain samples. Therefore, the used data set should be sufficiently large. The Eigen split [14] of the KITTI data set [16] is commonly used in MDE and consists of around 24.000 training samples. Since it can be argued that this amount is sufficient for training an MDE method, the requirement is set that the previously mentioned 10%-subset should be sized in the same order of magnitude. The used data set should therefore consist of more than 100.000 RGB-depth pairs. This requirement invalidates both KITTI [16] itself, as well as Make3D [47]. While Human Pose [11] might be sufficiently large, its lack of diversity in scenes and objects makes its use undesirable. NYU-Depth V2 [51] also has enough samples, but only consists of indoor scenes. Since some cues (especially linear perspective and texture gradient) are mainly present in outdoor scenes, this is not preferable for this research.

Alternatively to these recorded, real-world data sets, virtual data sets exist. Such data has the following advantages:

- **Accuracy/Density**: The depth is generated from ground-truth 3D data, so it does not suffer from a sensor that is inaccurate or can only (accurately) record depth up to a certain distance, leading to sparse depth data. Additionally, there is no disparity between a camera and a depth sensor, which could lead to more inaccuracies/missing data points. First of all, a complete depth map (no missing data points) is desirable because there are more points that can be used for a loss function. An additional advantage is that the depth map can be resized in the same manner as the RGB image, without zero values distorting the resulting depth map. Figure 8 in Appendix E shows a sparse and a dense depth map.
- **Cost/Scale/Speed**: Only capable computing hardware is required, without needing possibly expensive recording hardware. The latter possibly requires comparable computer hardware for post-processing anyway. Furthermore, due to the freedom of movement in a virtual environment, more diverse data can be generated in a shorter time frame. Finally, no manual labour is required for getting permission for recording sensitive data or filtering out those or otherwise undesirable samples.

Such virtual depth data sets exist: FlyingThings3D [38], Driving [38], Monkaa [38] and MVS-Synth [22]. FlyingThings3D and Monkaa use an unrealistic world environment and unrealistic object respectively, while cues are based on a real-world environment. Additionally, all four data sets are not sufficiently large for Experiment 2. Finally, Virtual KITTI 2 [6] is both large enough and uses a more realistic environment, however, its textures are noticeably less realistic than the contributed data set from this work. Figure 9 shows images from these virtual data sets.

Description For the data collection, a virtual environment resembling an American city[2] is used. To record the data, a virtual camera is placed at eye level and moved along a path while periodically recording a snapshot of both image

[1] https://www.unrealengine.com/en-US/unreal-engine-5

[2] https://www.unrealengine.com/marketplace/en-US/product/city-sample

data and the corresponding depth map. The path is repeated multiple times with a different yaw of the camera each time. Path instances are recorded using varying world settings: either the day or night setting, as well as different amounts of fog.

The depth values are stored as unsigned 16-bit integers (integers from 0 to 65535). By dividing these values by 256, their values in meters can be determined. This allows for a maximum value of almost 256 meters to be stored, which is why all values greater than that are replaced by this value. In the raw data, the recorded images are of size 480x853 (9:16), but for Experiment 2, these are centre-cropped to 480x480 and resized to 256x256.

The training set consists of seven paths, each repeated with approximately 10 yaw values, with various world settings for each instance. The length of each path is different but chosen so that the distance between each snapshot is approximately one meter. The resulting set consists of 113.759 RGB-depth pairs. The test set consists of a single path, recorded at a distinctly separated location, with 8 yaw values and various world settings. The resulting set consists of 773 RGB-depth pairs.

4. Methodology

This section describes the used MDE model that can incorporate cue channels, the methods for extracting and encoding cues from an image, and the experiments that are performed to answer the research questions.

4.1. Model

This work continues on the work of Al-jibouri [2], who used a cGAN according to the 'pix2pix' configuration [24]. There are several reasons why a cGAN can be advantageous over using a traditional CNN. First, instead of using a fixed loss function, a cGAN learns it during training. This makes it applicable to diverse tasks (e.g. image-to-image and image-to-depth) that would otherwise require diverse loss function [24]. Furthermore, cGANs have exhibited better generalisation (performance on unseen data) than traditional CNNs [29]. Finally, Chen et al. has shown that adversarial training can be beneficial for MDE [10].

Al-jibouri modified 'pix2pix' to produce a depth map instead of images. For this work, it was further modified to allow for cues to be extracted from an image and added into the input space. The generator consists of an encoder-decoder structure, where an image is progressively downsampled into a compact representation (bottleneck) before it is upsampled again to produce a depth map. Skip-connections are present between mirrored layers so that low-level information can be passed over the bottleneck. Therefore, the bottleneck does not need to represent the entire image, only information relevant for depth estimation. A more detailed description of the model can be found in Appendix A. First, the appendix provides a schematic that describes how a single RGB-depth pair is passed through the generator and discriminator. Furthermore, it describes the 'pix2pix' architecture and how it is trained. On top of that, it describes the modifications made by Al-jibouri to use the model for depth estimation, as well as the modifications made in this work. Finally, it describes the layers of both the generator and the discriminator.

4.2. Cue Extraction

This subsection describes for all depth cues incorporated in this work, how information related to them is extracted from a single image and encoded into channels that can be used in addition to the three channels of an input image. Complete implementations of each cue extraction method including the used parameters are publically available[3]. This subsection describes a more general overview of how each method works. A global overview including visual examples of the encodings of each method can be found in Figure 2. A summary of the methods can also be found in Table 2. The goal for the cue methods is to improve the depth estimation, while runtime optimisation is not considered yet. Therefore, some methods have such a long duration, that executing them 'live' during training is quite unpractical. Therefore, each cue method is applied to all images and the output is stored on disk. During training the cue method outputs are fetched along with their corresponding image.

Familiar size (F) To extract information about familiar size Mask R-CNN [21] is used to detect objects in an image. This method extends the object classification and bounding box generation of Faster R-CNN [43] with the prediction of a mask for every recognised object. It achieves this by the addition of a separate branch that predicts a mask for each object using image segmentation parallel to the existing branch of Faster R-CNN. The used configuration is built on the ResNet-50 backbone [20] and is pre-trained on the COCO data set [34] to predict the masks of 81 classes.

For each image, the Mask R-CNN is used to predict a mask for every object it recognises. To ensure an object is not classified twice as two different classes, only masks with a confidence score of greater than 50% are considered. For each class, manual estimations have been made for the average height, width and depth. The final resulting encoding is a 3-channel (HxWxD) array of the same shape as the input image. For each class-mask pair, each of the three dimensions of the class is used as the value in their respective channel for every pixel in the mask. Pixels that have not been classified all have value 0 for each channel.

[3] https://github.com/jochemroes/cues

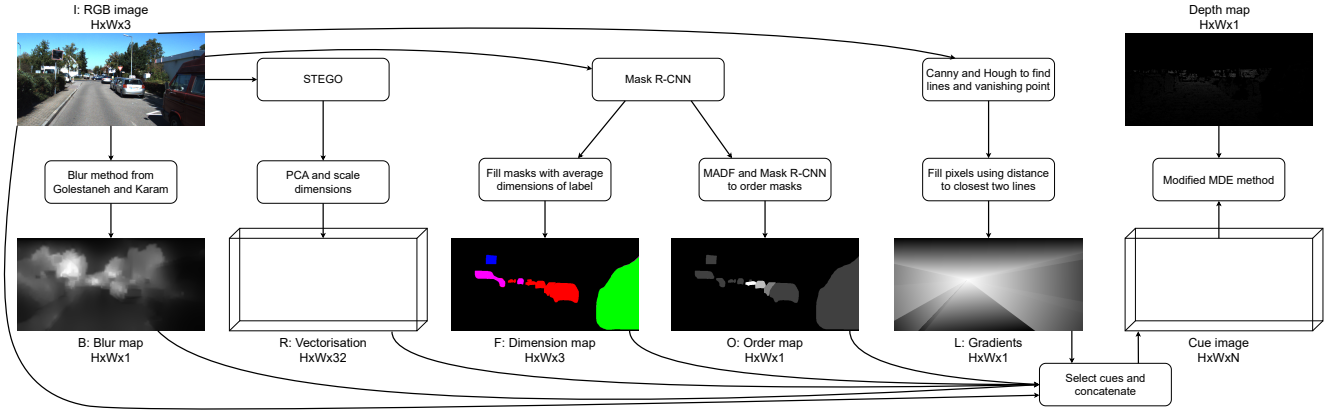

Figure 2. Global overview of all the cue extraction methods and how they are fed into an MDE method.

Overlap (O) The overlap method is based on the fact that the observable area of a partially obscured object grows if the object is not obscured anymore. For this method, the same Mask R-CNN is used. Additionally, MADF [60] is used for image inpainting. MADF predicts the missing pixels of a masked-out region of an image. An MADF configuration is used that is trained on the Places2 data set [59].

The overlap method determines the object masks in the same way as familiar size. For every pair of masks with overlapping bounding boxes, the MADF model is used to generate two new images. These two images are the result of inpainting either mask. Mask R-CNN is applied to both images, and the size of the mask that has the highest overlap with each original mask is determined. One mask growing significantly in the image mask where the other mask is inpainted means it is further away. For the pair, this results in a directed edge ab, representing object a obscuring object b.

This information is encoded into an image channel using a directed graph. Every vertex v represents an object mask and initially has depth value $d_v = 1$, children $C_v = \emptyset$ and offspring $O_v = \emptyset$. The edges determined previously are added to the graph using the procedure add:

```
PROCEDURE add(a, b)
    IF a ∉ O_b
        C_a ← C_a ∪ {b}
        O_a ← O_a ∪ {b} ∪ O_b
        update(b, d_a)

PROCEDURE update(v, d)
    IF d ≥ d_v
        d_v ← d + 1
        FOR ALL u ∈ C_v
            update(u, d_v)
```

The $a \notin O_b$ condition prevents the algorithm from infinitely iterating in case that some edges form a directed cycle. This is likely an error due to the rarity of an object obscuring an object that is placed in front of it. The final encoding is a single sparse image channel with d_v as the value for the pixels in the mask of every vertex v.

Relative size and foreshortening (R) STEGO [19] is used to extract information related to relative size and foreshortening. This is an image segmentation method that requires no labelled data during training. It is based on self-supervised feature learning, where a model learns spatially-varying features for an image, i.e. a vector for every pixel. These vectors are pooled into global features for a region. During training, STEGO learns to transform regions from the same or similar images into similar representations and regions from two diverse images into diverse representations. After training, the global features are discarded and only the vectors for each pixel are used. If two pixels have similar vectors, they likely originate from the same class. Using cosine similarity and K-Means clustering, the pixels are clustered to generate an image segmentation [19]. For this work, STEGO pre-trained on the COCO data set was used [34].

Such an image segmentation of the image allows for identifying which objects are similar to each other and how they differ in size. However, preliminary experimentation with these segmentations did not generate satisfactory results. Therefore, the clustering layer from STEGO is removed, so that the pixel vectorisations are used as the final encoding. These vectors already contain all the information about how similar pixels are to each other, and therefore how likely they belong to the same object (relative size) or subobject (foreshortening). The resulting encoding for relative size is similar to that of Auty and Mikolajczyk. However, a pixel vector is not limited by the predicted class of a supervised image segmentation method, so each individual vector should be more informative. To decrease storage size, the pixel vectors are transformed from size 90 to 32 using principal component analysis [40], after which the

values in each dimension are scaled according to 8-bit unsigned integers.

Blur and texture gradient (B) Encoding blur information consists of producing a blur map. This is a single channel of the same height and width as the input image, but the pixel values are sharpness scores. If an image contains a uniform texture that is not influenced by blur too much, a valid blur map contains a gradient directly related to the depth of the texture. The method from Golestaneh and Karam produces such a blur map. For multiple scales of the image, the method takes a small patch around each pixel and uses the discrete cosine transform to convert the image patch to the frequency domain. In this frequency domain, sharper images generate greater amplitudes for high frequencies than less sharp images do. Therefore, the high-frequency amplitudes are used to generate the pixel sharpness score. The sharpness scores are normalised from 0 to 1 [18].

Linear perspective (L) To extract linear perspective Canny edge detection [7] followed by Hough line detection [23] is used to detect line pieces/lines corresponding to linear edges in an image. The point with minimal distance to each line is deemed to be the vanishing point. After this, The lines are filtered on them (nearly) intersecting with the vanishing point and are redrawn with one end at the vanishing point and one end at the image border. Then, lines that lay particularly close to each other are merged. Lines that result from significantly fewer merges than the greatest merge count seen are filtered out. Finally, the vanishing point laying outside of the image, fewer than four lines remaining, or no lines being merged results in the image being deemed to show too little linear perspective. In this case, the vanishing point is replaced by the centre of the image, and the lines are replaced by four lines each from the vanishing point to one corner of the image.

To encode this extracted information into an image channel, a value is determined for every point according to the summed distance to both lines it lies between. These are then normalised according to the maximum value in the result. After subtracting these values from 1, the resulting channel consists of gradients shaped like a triangle, where the vanishing point has the greatest value (1), and the zero-values are at the border of the image. If the edges of such a triangular gradient correspond to the edges of a texture, then the gradient is directly related to the depth of the texture.

For images that show linear perspective, the generated channel could function as a low-resolution estimation of the depth which is then improved using the image data by the model. Eigen et al. has shown that determining a low-resolution depth map, before refining it using the image data, can be useful for generating depth maps [14].

4.3. Experiments

This subsection describes the experiments that are performed in order to answer the research questions.

Data sets Multiple data sets and train/test splits are used in the experiments. The first is KITTI [16] (further described in Appendix E) in two splits. *Kitti* is the original split from Geiger et al. of around 57K/23K. *Eigen* is the split from Eigen et al. of around 24K/693. Additionally, the data set from Section 3 is used in the splits $\mathit{Virt}_{10} \subset \mathit{Virt}_{25} \subset \mathit{Virt}_{100}$. The subscripts represent the percentage of the training data that is used, where Virt_{100} contains around 113K training images. The subsets are randomly sampled to approximate the same diversity of Virt_{100}. All three splits use the same 773-image test set.

Metrics The following metrics are used for evaluation (which are commonly used in the relevant literature [30, 14, 2, 5, 33]): **thr**esholded difference, **abs**olute relative difference, **squ**ared relative difference, root mean square error (**rms**) and logarithmic root mean square error (**lms**). These are further described in Table 1. There are two ways of determining the values, both of which have been used in related work [26, 33]: either treating all pixels in a test set as one pixel collection or determining each metric separately for each image before determining the mean value. Since the former prioritises images with more depth measurement this used for all experiments that use the model from Subsection 4.1. However, the latter is used for the state-of-the-art experiments, to minimise the differences with those methods. Specific to this work, is the **flu**ctuation index, which is based on standard deviation and determines the relative difference between the evaluation values of model snapshots and a moving average (of length 5). This results in a metric that describes the instability of a model during training. Finally, the **exp**eriment comparison index is introduced, which summarises how two results compare over the previous metrics. These two metrics are also described in Table 1.

0) Cue usefulness For each cue, an experiment is performed with the channel(s) from the extracted cue as the only input of the model. These are trained for a single epoch and evaluated on *Kitti*. The baseline (N) is trained in the same manner as the cues, but it only contains data where every pixel in the input has value 100. Therefore, the input data provides no information and the model is only able to learn a general bias for the depth estimation. This experiment determines whether each cue method provides information to the model that is relevant for depth estimation and merits further exploration.

ID	Formula	Unit
thr	$\frac{\lvert\{(p,\hat{p})\in\mathcal{P}\lvert\delta_i\}\rvert}{\lvert\mathcal{P}\rvert}; \delta_i = \left[\max(\frac{p}{\hat{p}}, \frac{\hat{p}}{p}) < 1.25^i\right]$	-
abs	$\frac{1}{\lvert\mathcal{P}\rvert}\sum_{(p,\hat{p})\in\mathcal{P}} \frac{\lvert p-\hat{p}\rvert}{p}$	-
squ	$\frac{1}{\lvert\mathcal{P}\rvert}\sum_{(p,\hat{p})\in\mathcal{P}} \frac{(p-\hat{p})^2}{p}$	m
rms	$\sqrt{\frac{1}{\lvert\mathcal{P}\rvert}\sum_{(p,\hat{p})\in\mathcal{P}} (p-\hat{p})^2}$	m
lms	$\sqrt{\frac{1}{\lvert\mathcal{P}\rvert}\sum_{(p,\hat{p})\in\mathcal{P}} (\ln p - \ln \hat{p})^2}$	ln m
flu	$\frac{1}{\lvert\mathcal{M}\rvert}\sum_{m\in\mathcal{M}} \sqrt{\frac{1}{\lvert\mathcal{X}_m\rvert-1}\sum_{x\in\mathcal{X}_m} \frac{(x-\mu_x)^2}{\mu_x}}$	-
exp	$\frac{1}{\lvert\mathcal{M}\rvert}\left(\sum_{m\in\mathcal{M}_\uparrow} \frac{m_a}{m_b} + \sum_{m\in\mathcal{M}_\downarrow} \frac{m_b}{m_a}\right)$	-

Table 1. The evaluation metrics defined using a pixel collection $\mathcal{P}$ consisting of ground truth depth values p paired with estimated values $\hat{p}$. For **flu**, $\mathcal{M}$ represents a collection of any combination of the previous metrics that are used for evaluation. $\mathcal{X}_m$ is the collection of scores resulting from evaluating all snapshots of a model on metric m. μ_x is the mean score of the values generated by a window of snapshots where the snapshot that generated x is in the centre. For **exp**, $\mathcal{M}_\uparrow$ and $\mathcal{M}_\downarrow$ are the metrics for which higher and lower scores respectively are preferred. m_a is the score for a metric m for experiment a that is compared to the metric score of experiment b (m_b).

1a) Final performance For each 'useful' cue, an experiment is performed where the input consists of the original image data concatenated with the channel(s) produced by cue methods. The baseline model only takes the image data as its input. The model from Subsection 4.1 is trained for 50 epochs on *Eigen*, before being evaluated on its test set. This allows for studying the influence of each added cue method on the final performance. Additionally, it allows for studying if combining the cue methods has any additional effect on the model. This experiment corresponds to **RQ1**.

1b) Training efficiency The same data set and (combinations of) cues from Experiment 1a are used for this experiment. Instead of only evaluating the final performance, each model is evaluated at a fixed interval during training. This allows for studying both the influence of cues on the training efficiency of a model and the stability of the training process. A stability issue of the original model (further described in Section 6 limits the ability to study training efficiency. Therefore, BinsFormer is used for this experiment. Its standard configurations are used [33], except that the encoder is not pre-trained and that the random image augmentations layers are removed. Every 800 iterations (i.e. 6400 images), each model is evaluated on the test set. Like Experiment 1a, this experiment corresponds to **RQ1**, while also being related to **RQ4**, since BinsFormer is a state-of-the-art method.

2) Data efficiency While the results of Experiment 1 should already determine the effect of added cues on the change in model performance during training, they are limited by the amount of available data. Regardless of the number of epochs and iterations, if the amount of training data is limited, the maximum performance achieved during training will be limited as well. For this experiment, Both the baseline model and the model that contains all successful cues are trained for 5.000.000 iterations, with snapshots being saved at every 100.000 iterations, which are all evaluated on the test set of the virtual set. For each metric, the average of the five best scores seen is determined. These experiments are repeated for every subset of the virtual set: *Virt*$_{10}$, *Virt*$_{25}$ and *Virt*$_{100}$.

The results of this experiment allow for studying the influence of cues on achievable performance and how this influence changes when more data is available (**RQ3**). Additionally, it provides information for determining the differences in the amounts of data needed to achieve comparable performance between the baseline and models that utilise cues (**RQ2**).

3) State-of-the-art Three state-of-the-art MDE methods are used for this experiment: BTS [32], AdaBins [5] and BinsFormer [33]. While each method works differently, they all use an encoder. This encoder transforms images into a different representation, which is used to create the depth map. For this experiment, these encoders are modified to use N image channels instead of 3. Every method also uses a pre-trained configuration of the encoder. These pre-trained weights provide a significant performance increase. For the first layer, the weights are repeated so that more than 3 image channels can be used. Besides these modifications, each method is used in its original configuration.

The original configurations include random image augmentations, like random crops and rotations. This leads to uncertainty in terms of the exact input images. Due to the long runtimes of some of the cue methods, the cue encodings for training images are calculated before training. The random augmentations make it impossible to do this for each exact input image. Therefore, the cue methods are only performed on the original images, and during training the same rotations and crops are applied. This could lead to a slight, unfair advantage, due to the cue methods being able to use parts of an image that the MDE method cannot. On the contrary, for relative size, this approach is likely disadvantageous, since STEGO resizes its inputs to 320x320. Cropping an image before passing it through STEGO would lead to a more informative encoding than cropping the encoding of the original image.

Each method uses $IRFO$ input, which is evaluated on *Eigen* before being compared to their original results.

5. Results

Experiment 0 The results in Table 4 show that both linear perspective (L) and blur/texture gradient (B) considerably outperform the baseline (input consisting of only one value, N) after training for a single epoch. The results from the other cue methods are omitted due to their positive results in Experiment 1, making their Experiment 0 results irrelevant.

Experiment 1 The same table shows that neither linear perspective (IL), nor blur (IB) is able to noticeably improve the performance of the baseline model (I). Both overlapping (IO) and familiar size (IF), as well as relative size (IR), all considerably outperform the baseline for every metric used. IF seems to slightly outperform IO, while both are noticeably outperformed by IR. $IRFO$ outperforms all other models, generating the best scores seen in Experiment 1a, for every metric.

Figure 3 shows how the **squ** performance of each model performs relative to the baseline model during training. Since **squ** is an error metric and since the baseline is used as the numerator in the normalisation process for creating the figure, greater values correspond to better performance. Since the baseline is a growth curve, greater values at the beginning of training correspond to greater training efficiency, while greater values at the end of training correspond to better final performance.

With this in mind, it is apparent how every (remaining) individual cue method allows for a substantial increase in training efficiency. Although IF is slightly worse than the baseline at the very earliest stage of training, it achieved a performance increase of more than 50% only slightly later. Furthermore, every individual cue method allows for a decreased **flu** score, i.e. an increase in the stability of training.

When combining all cues ($IRFO$), there is still a sizable increase in training efficiency compared to the baseline. However, it does not noticeably improve further upon the individual cues. Finally, combining the cue methods that originally increase the stability, does not cause a further increase, but leads to the stability not being better than the baseline.

Experiment 2 To interpret the results for this experiment in Table 4 in terms of data efficiency (**RQ2**), three subexperiments are relevant: $IRFO(\mathit{Virt}_{10})$, $IRFO(\mathit{Virt}_{25})$ and $I(\mathit{Virt}_{100})$. These results together with the summary in Table 5 show that training a cue model (a model that uses all three successful cues) on 10% of the used training set already leads to performance on the test that is worse than the baseline model using 100% of the training data. However, when this amount is increased from 10% to 25% the same baseline is outperformed for most metrics, and a summarised **exp** score of 1.00 is generated. This means that the cue model using 25% of the training data generates about the same performance as the 100% training data baseline model.

For studying the effect of increasing the amount of data (**RQ3**) every pair of I and $IRFO$ trained on the same amount of data is to be compared. Table 5 is especially relevant for this. For Virt_{10}, Virt_{25} and Virt_{100} the **exp** scores are 1.12, 1.09 and 1.08 respectively. For Virt_{10} the advantage of using cues is the greatest, and for Virt_{100} it is the smallest. However, the differences in **exp** scores are rather small, and for Virt_{100} cues still allow the model to outperform its baseline for every metric.

Experiment 3 As can be seen in Table 4 and Figure 4 the results of Experiment 3 do not mirror those of Experiment 1. Adding cues to the input causes lower performance at every stage of training for the state-of-the-art methods (when using pre-trained encoders). The only observed positive effect is the increase in stability for BinsFormer.

On the contrary, when no pre-training is used for BinsFormer (BFn), the positive effects of adding cues are present. Table 4 as well as Figure 4 show that for BFn the addition of cues has a positive effect on the performance.

ID	Method	Encoding	Chan's
I	Identity	RGB image	3
N	None	Blank image	1
R	Relative size	Vectorisation	32
F	Familiar size	HxWxD map	3
L	Linear perspective	Gradient	1
O	Overlap	Order map	1
B	Blur	Sharpness map	1

Table 2. Summary of the cue extraction methods. **Method** refers to a cue method that extracts information from an RGB (N produces the same blank image for every input image). **Encoding** refers to what the extracted information is encoded as. **Channels** refers to the number of image channels used by that encoding.

ID	Name	Authors	Ref
Eig	-	Eigen et al.	[14]
Jib	-	Al-jibouri	[2]
BTS	Big-to-Small	Lee et al.	[32]
AB	AdaBins	Bhat et al.	[5]
BF	BinsFormer	Li et al.	[33]
BFn	*BinsFormer without pre-trained encoder*		
UD	URCDC-Depth	Shao et al.	[49]
MIM	SwinV2-L 1K-MIM	Xie et al.	[55]

Table 3. List of MDE methods that are used in one of the experiments, or for which the results are sourced for Experiment 3.

Experiment settings			Thresholded difference (↑)				↓	↓	↓	↓
MDE	Input	Split	$\delta_{1/2}$	δ_1	δ_2	δ_3	abs	squ	rms	lms
Jib	N	*Kitti*	.280	.365	.459	.587	.373	4.60	12.4	.683
Jib	L	*Kitti*	.544	.698	.835	.910	.194	2.30	9.24	.363
Jib	B	*Kitti*	**.568**	**.737**	**.885**	**.950**	**.163**	**1.57**	**7.56**	**.292**
Jib	IB	*Eigen*	.700	.870	.966	.990	.110	.669	4.39	.176
Jib	IL	*Eigen*	.694	.872	.967	.990	.110	.658	4.40	.176
Jib	I	*Eigen*	.703	.876	.967	.990	.108	.653	4.35	.174
Jib	IO	*Eigen*	.715	.882	.971	.993	.103	.583	4.20	.163
Jib	IF	*Eigen*	.718	.884	.972	.993	.101	.569	4.15	.162
Jib	IR	*Eigen*	.734	.892	.975	.994	.096	.526	4.05	.156
Jib	$IRFO$	*Eigen*	**.741**	**.894**	**.979**	**.995**	**.093**	**.494**	**3.90**	**.150**
Jib	I	$Virt_{10}$	.473	.845	.972	.988	.155	1.86	11.0	.212
Jib	I	$Virt_{25}$	.563	.893	.978	.991	.129	1.40	9.76	.185
Jib	I	$Virt_{100}$	.619	.893	.982	.993	.114	1.08	8.73	.171
Jib	$IRFO$	$Virt_{10}$	.583	.894	.972	.982	.129	1.55	10.2	.177
Jib	$IRFO$	$Virt_{25}$	.631	.912	.983	.991	.110	1.16	9.10	.165
Jib	$IRFO$	$Virt_{100}$	**.692**	**.926**	**.986**	**.993**	**.094**	**.970**	**8.54**	**.151**
Eig	I	*Eigen*	-	.702	.898	.967	.203	1.55	6.30	.282
BFn	I	*Eigen*	.720	.901	.979	.994	.101	.496	3.45	.140
BFn	$IRFO$	*Eigen*	.732	.908	.984	.996	.096	.457	3.36	.133
BTS	$IRFO$	*Eigen*	.824	.942	.991	.998	.068	.264	2.76	.105
AB	$IRFO$	*Eigen*	.824	.950	.993	.999	.069	.246	2.70	.102
BTS	I	*Eigen*	-	.956	.993	.998	.059	.245	2.76	.096
BF	$IRFO$	*Eigen*	.850	.956	.994	.999	.062	.213	2.42	.095
AB	I	*Eigen*	-	.964	.995	.999	.058	.190	2.36	.088
BF	I	*Eigen*	-	.974	.997	.999	.052	.151	2.10	.079
UD	I	*Eigen*	-	.977	.997	.999	.050	.142	2.03	.076
MIM	I	*Eigen*	-	**.977**	**.998**	**1.00**	**.050**	**.139**	**1.97**	**.075**

Table 4. Results from Experiment 0, 1a, 2 and 3. Each MDE method ID is described in Table 3. **Bold** MDE IDs means the results are sourced from the literature. The input IDs refer to concatenated methods from Table 2. **Bold** metric scores are the best, underlined the second.

		$IRFO$		
		10	25	100
I	10	**1.12**	1.24	1.35
	25	.991	**1.09**	1.18
	100	.915	1.00	**1.08**

Table 5. Results from Experiment 2 summarised using the **exp** metric. The row and column title values refer to the percentage of *Virt* used for each experiment. The $IRFO$ models act as experiment a, which is compared to the I models (experiment b), so a greater score represents an $IRFO$ model outperforming the I model it is compared to. The underlined values are relevant for data efficiency (**RQ2**), while **bold** values are relevant for **RQ3**.

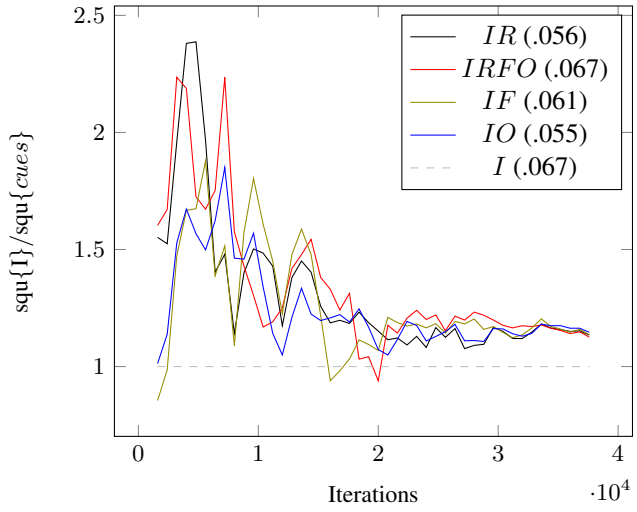

Figure 3. Experiment 1b (training efficiency). Greater scores correspond to better performance. Scores result from the mean **squ**-score of a window of three snapshots for both the baseline I and each model that uses cues. For visualisation purposes, the former is divided by the latter. The legend entries contain the **flu** scores.

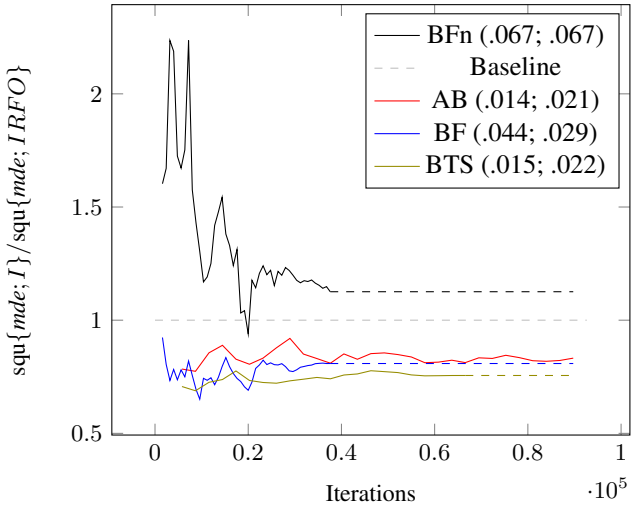

Figure 4. Same set-up as Figure 3 for the models in Experiment 3. The MDE methods are described in Table 3. The baseline is different for each MDE method, namely that method with I input. The legend entries contain the **flu** scores in comparison to the baseline $(I; IRFO)$.

6. Discussion

Linear perspective Linear perspective considerably outperforms the baseline in Experiment 0, albeit with a smaller difference than B. This indicates that the data extracted by the linear perspective method contains relevant information for depth estimation. Additionally, the linear perspective constructs an image from scratch using information from the input image, so it is not just a simple mutation of the input image (e.g. conversion to grey-scale). This fact supports the possibility that the linear perspective data is informative on top of the input image.

However, Experiment 1 has shown that this is not the case, since the addition of the linear perspective data does not positively affect the depth estimation. One likely reason is that the linear perspective cue only provides insight into a global estimate of the relative depth. As shown by the near-perfect δ_3 metric results, this is not something that MDE methods particularly struggle with. Therefore, the benefit of adding linear perspective is likely minimal. Combined with the fact that parallel edges (and linear perspective) are not omnipresent, the linear perspective method might simply not have any value for the method. Lastly, the method can be prone to inaccuracies, for example, because of incorrect vanishing point detection. Incorrect cue information leads to an incorrect representation of the estimated depth.

There are still many improvements or different methods that could be included. Examples include support for more than one vanishing point, different ways of encoding the extracted information, and inclusion of a confidence measure. However, because linear perspective is not (sufficiently) present in many images and because the results do not show any sign of performance improvement, it is decided to not explore linear perspective further.

Blur On the one hand, the method from Golestaneh and Karam consists of a large amount of non-linear operations. Therefore, a blur map could contain information that an MDE method is not able to infer from the image data itself. On the other hand, if that is the case, the blur map still needs to be both accurate and informative enough for the estimation of depth. This does not seem to be the case since no improvement is observed in Experiment 1.

For blur to be of use, the image needs to contain enough differences in sharpness. While differences in sharpness can be very clear in a professional photograph, they might be less clear in the images of the KITTI data set. Therefore, the blur map might not provide enough additional information to the MDE methods. Furthermore, if blur is not sufficiently present in the image, the method could be more useful for texture gradient. However, similarly to linear perspective, this information might simply not provide any information that cannot be extracted from an RGB image. Finally, the sharpness values might be inaccurate due to shortcomings of the method. For example, a uniform texture positioned near the focal point might be falsely identified as being blurred.

While alternative methods could be used to extract blur and/or texture gradient information, these would likely also suffer from the previously described issues, which is why blur and texture gradient are not further explored.

Familiar size The increase in training efficiency can be explained by the fact that determining depth from the approximate dimensions of an object likely requires fewer operations than determining it from image data. Therefore, learning these operations should require fewer training iterations. Additionally, the MDE methods do not need to learn the position and shape of salient objects. Besides training efficiency, training stability is also increased. A possible explanation for this is that the familiar size data is more uniform than the image data. Less variety in the input data could lead to less variety in the estimation. Furthermore, to replicate the steps from the familiar size method, an MDE method would need to learn object recognition. It might not have enough training data and enough capacity to learn to extract this information aside from the other steps needed for depth estimation. While replication of the method is likely not needed to match or outperform the performance, it can explain why there is a substantial performance difference after finishing training.

A shortcoming of the familiar size method is the sparsity of its output. All objects present in the image that are not recognised are not included in the final encoding. Additionally, the names of predicted classes are a bottleneck for the method. An example of why this can be problematic is the class name 'truck', which could both refer to a 'pick-up truck' and a 'semi-truck', two objects with great differences in dimensions. Similarly, the chosen dimensions for each class might not be accurate for each instance of that class (e.g. instances of the class 'car' can exhibit significant variation in their dimensions). Incorrect dimensions can lead to incorrect depth predictions. All of these issues can be minimised by incorporating an object recognition method that includes more diverse classes and more specific classes (e.g. one that supports both 'pick-up truck' and 'semi-truck' as classes, instead of only 'truck'). However, such a method has its own drawbacks, which are further discussed in Subsection 7.1.

Overlap The results of overlap mostly mirror those of familiar size, albeit slightly subpar (except for the greater instability). The improvement of the training efficiency and final performance could be explained by the provided position and shape of salient objects. For overlap, this feature of the data is more expressive than it is for familiar size,

since neighbouring objects of the same class have different values. By receiving this information as input, the MDE method does not need to learn it itself, which frees up capacity for further improving the depth estimation. The reason that it performs subpar to familiar size is likely that familiar size does not just contain information about position and shape, but also the approximate dimension, which should be more meaningful for depth estimation than the ordering overlap provides. The stability improvement could be explained by the increased uniformity of the input data, as it is for familiar size.

Like familiar size, the overlap method is also limited by the number of classes the object recognition is trained on. Unlike familiar size, overlap does not suffer from any inaccuracies due to ambiguity or non-specificity of class names, since only the object masks are used.

Relative size The relative size (and foreshortening) method causes substantial increases in final performance and training efficiency. Both can be explained by the fact that in a human-analogous way, the MDE method utilises prior knowledge about the similarity of (parts of) objects. In combination with the observed object sizes, this prior knowledge allows for estimating depth. Because an MDE method does not need to learn to extract this from the image data, it is able to generate better performance at an earlier stage. The better final performance can be explained by the fact that this knowledge cannot be learnt from the image data. This information is so useful, that the cue method can outperform the other ones. Although the fact that relative size produces dense data, unlike overlap and familiar size, might also have a significant effect. A possible explanation for the increased stability could be that STEGO produces data that is more uniform than the input data. While colours within a single image may vary significantly, STEGO is trained to produce similar values for pixels within a single object.

The relative size method has an additional advantage over familiar size and overlap: it uses unsupervised training. This could mean that depth estimation can be further improved by utilising a method that does not rely on any sort of labelled data. One of the main hurdles of monocular depth estimation is that it requires large data sets of accurate ground truth depth maps, which can be hard to obtain [17]. The relative size vectoriser only requires images, which are widely available. New depth estimation methods could therefore consist of a vectoriser and a depth estimator. The vectoriser can then be trained unsupervised on a large collection of images, including the input images from the depth data set. After this, the depth estimator can be trained using the ground truth depth data and the images with vectorised pixels.

Combining cues By combining all three successful cue methods ($IRFO$), the best final performance is achieved. Familiar size produces the same value for neighbouring objects of the same class, which could be misinterpreted as them being a single object. Overlap uses the same masks as familiar size, but produces separate values for neighbouring objects. Adding overlap to familiar size could therefore mitigate that problem and improve performance. Appendix D contains results that support this. Additionally, since these two methods work quite differently than relative size, they might contain knowledge that is not present in the relative size encodings. Finally, although more useful data is present in each image, the data also has a greater dimensionality, complexity and variability. This could explain why the training efficiency is not noticeably improved and why the stability is decreased.

Data efficiency The results show that the model using cues requires considerably less data to reach the same performance as the model without cues. For every training instance, more data that is relevant for depth estimation is present. This gives the model more insight into the depth of each instance. A model requiring fewer data for training enhances the usability of the model, since it can reach acceptable performance on a greater amount of data sets.

More training data The results from Experiment 2 show that increasing the amount of available training data decreases the effect that the cues have on performance. However, when all of the training data is used a sizable difference remains, with no indication of this difference being eliminated when even more training data would be present. First of all, this can be attributed to the limited capacity of the model. Cues provide a way of 'out-sourcing' capacity from the model to the cue extraction methods. Therefore, using cues provides an advantage that cannot be overcome by using more training data. Repeating the same experiments for a model with greater capacity would likely result in less substantial differences. But such a model is still limited by the information present in the depth data set it is trained on. The cues are trained on different data sets, which contain information unseen in the used depth data set. Given that this information is relevant for depth estimation, the benefits of using cues should remain.

State-of-the-art Applying the three successful cues to state-of-the-art methods does not improve performance in any way. Although the stability is improved for BinsFormer, it is worsened for AdaBins and BTS, making that result irrelevant. On the other hand, for Experiment 1b, BinsFormer without pre-training was positively influenced by the cues.

This means that cues can are able to improve state-of-the-art methods, but they are in conflict with the methods used for pre-training the encoders of those methods. Similar to cues, the pre-training methods improve depth estimation by incorporating knowledge that the MDE method cannot learn from its data set. They provide an increase in depth estimation that cannot be matched by just using cues. But the pre-training is heavily biased towards only image inputs, leading to a decrease in performance when both cues and pre-trained encoders are used.

The cues are an alternative way of improving depth estimation. While currently, the encoder pre-training achieves better performance, the cue methods have a lot of room for optimisation (see Subsection 7.1). Furthermore, an advantage of cues over encoder pre-training, is that different methods can easily be concatenated. The knowledge from multiple data sets and different methods all be combined into one input image. Finally, using cues and a pre-trained encoder are not necessarily mutually exclusive. While the current pre-trained encoders are all biased towards 3-channel images, an encoder could also be pre-trained specifically for N-channel images that contain cue information. In fact, masked image modelling (MIM) [55] is a pre-training method that is not dependent on a specific number of image channels, while also having led to the best-known MDE performance on *Eigen* [55]. The resulting MDE method would benefit from both the cues and the encoder pre-training. Future work consists of creating such a method, which is further described in Subsection 7.1.

Model instability The reason why BinsFormer is used for Experiment 1b instead of the model described in Subsection 4.1, is that the latter suffers from instability problems when the relative size is used. When this is the case, the model does not produce anything meaningful for many iterations until at some arbitrary point its depth estimations improve, making a comparison of training efficiency impossible. One likely reason for this is that due to the greater amount of channels and possibly greater complexity of relative size, the generator takes longer to learn depth estimation. The discriminator could then quickly learn to distinguish between real and fake depth maps causing a high loss for the generator. After big changes to the weights at every iteration, at some arbitrary point in training, the generator does learn to fool the discriminator, and training can continue as usual. While not having a noticeable effect on the final performance, the training efficiency is greatly negatively affected. Due to this issue, the use of a cGAN might not be desirable for MDE when cues are used.

7. Conclusion

This work has shown that the addition of biological depth cues to the input space has many positive effects on monocular depth estimation. Three out of five incorporated cue methods cause a substantial increase in the final performance. All three also increase both the training efficiency and stability during training. Adding all these three methods to the input space (hereinafter called cue model) results in an additional increase in the final performance. Furthermore, the cue model requires much less data. On the used data set, the cue model with 25% of the training data achieved about the same performance as the baseline using 100% of the training data. Finally, while the utilisation of more training data decreases the effect of the cues, a substantial increase in performance is still present.

The reason for this is that cues still provide a way of 'outsourcing' model capacity that ensures an advantage over a non-cue model. While an increase in capacity could decrease the benefit the cues provide, such a model is still limited by what it can learn from the data by available depth data sets. The cue methods can be trained on different data, providing information that is not present in the depth data set used. This is especially beneficial for relative size, since it uses unsupervised training, leading to a method that can improve depth estimation, while only needing images for training.

RQ1) Effect of biological cues on MDE All cue methods showed that they produce useful information for depth estimation by outperforming the baseline N in Experiment 0. However, Experiment 1 showed that adding blur or linear perspective to the image input does not improve the depth estimation. The other cue methods show that they produce information that the model used cannot learn from the image data. Including familiar size, overlap or relative size in the input leads to a substantial increase in final performance and training efficiency. Combining all three cue methods further increases final performance.

RQ2) Data efficiency The prior knowledge present in the cue methods ensures that more relevant information is present in each training instance, and consequently that the model needs fewer instances to achieve the same performance. For the data set used, using only 25% of the training data with cues leads to similar performance to using all the training data without cues.

RQ3) Effect of an increase in data The benefits provided by using cues are most prevalent when smaller amounts of data are used. While increasing the amount of data does decrease the positive effects the cues have on the final performance, there is still a substantial difference between the model using cues and the baseline, with no indication of them reaching equal performance when even more data would be available. Therefore, the used model does not have the capacity to learn to diminish the advantages of

the prior knowledge provided by the cues. A model with a greater capacity would still be limited by the information present in the data set it is trained on. The cue methods can be trained on other data sets, leading to prior knowledge that cannot be learnt from the used depth data set.

RQ4) Comparison against state-of-the-art The benefits of using cues have not been reproduced when applying them to state-of-the-art methods. However, when the image encoder of a state-of-the-art method is not pre-trained, the cues do improve performance. This is because currently, these methods use pre-trained image encoders that are biased towards RGB images. While currently, the cues are in conflict with this, a future method could use an encoder that is pre-trained using images with cues. Such a method would benefit from both the cues and the encoder pre-training. This is part of future work.

7.1. Future work

One goal of future work is to outperform the state-of-the-art by using cues. To achieve this, a state-of-the-art MDE method will be used. The encoder is to be specifically pre-trained for these cue methods. The best-known performance on *Eigen* was achieved using BinsFormer [33] where the encoder was pre-trained in an unsupervised manner using masked image modelling (MIM) [55]. Since MIM is not dependent on a certain amount of image channels, a future setup could work as follows: cue methods are trained using their own data set; the frozen cue methods are applied to the images used to pre-train the encoder using MIM; the trained encoder is used for BinsFormer; finally, the frozen cue methods are applied to the images from an MDE data set, which are used to train BinsFormer to produce depth.

Furthermore, in terms of cues, a broad approach was taken for this work: every known cue that is in line with MDE requirements and operates at a relevant depth scale is included. It has been established that further exploring linear perspective and blur is not deemed advantageous, while familiar size & overlap and relative size show great potential. Therefore, future work includes the further improvement of these cue methods, with the goal of further improving MDE, while also improving the runtime performance so that the cue methods can be executed 'live' during training.

Familiar size & overlap The main limitation of the familiar size and overlap methods is the number of classes that are used by the object recognition method. Both cue methods produce sparse data. Additionally, for familiar size, the dimension values are prone to inaccuracies due to ambiguous class names and the average chosen dimensions being incorrect for an instance. All of these issues can be minimised by incorporating an object recognition method that can predict more classes, but such a method is prone to an increased estimation error [4].

An alternate way is using an end-to-end method that takes in an image and directly predicts the approximate height, width and depth, as well as the overlap ordering. While inaccuracies in the method could still lead to inaccuracies in the output, it would not suffer from the previously described issues. For training this method, virtual data can be used, similarly to the *Virt* data set. Instead of getting the depth from every pixel, the dimensions and ordering of the object of every pixel are gathered. The dimensions are present in the 3D model used for every object in a produced image, ensuring that the ground truth data is actually correct. Some fine-tuning on real images might be necessary, but by training most of the methods on virtual data, relatively little manual labour is required.

Relative size & foreshortening Improving the relative size method mostly consists of the creation of an image vectoriser where the focus does not lie on image segmentation but on the usefulness of the vectors for depth estimation. The method can be based on STEGO [19], but a more efficient encoder-decoder architecture, variable input size and a different number of output channels might be preferable. The training still consists of unsupervised training, taking advantage of the omnipresence of unlabelled images to gain knowledge about pixel similarities.

Appendix A. Model Details

Figure 5 shows an overview of the MDE model. The remainder of this Appendix describes the 'pix2pix' architecture, as well as the changes made to it by Al-jibouri, and the changes made for this work. A detailed description of the layers for both the generator and discriminator is provided as well.

Generator As mentioned at the beginning of Section 4, this work continues on the work of Al-jibouri [2], which used a cGAN according to the 'pix2pix' configuration [24]. In this configuration, the 'U-Net'-based generator consists of convolutional layers that progressively downsample an input image into more compact representations, until a 'bottleneck' is reached, which is progressively upsampled until the output has the same shape as the input image [44]. Additionally, skip-connections are present between mirrored layers, which stack the output of those pairs of layers. This allows for low-level information to pass over the bottleneck. Therefore, the bottleneck is still incentivised to contain information relevant to the image conversion but does not need to represent the entire input image, as a standard encoder-decoder would.

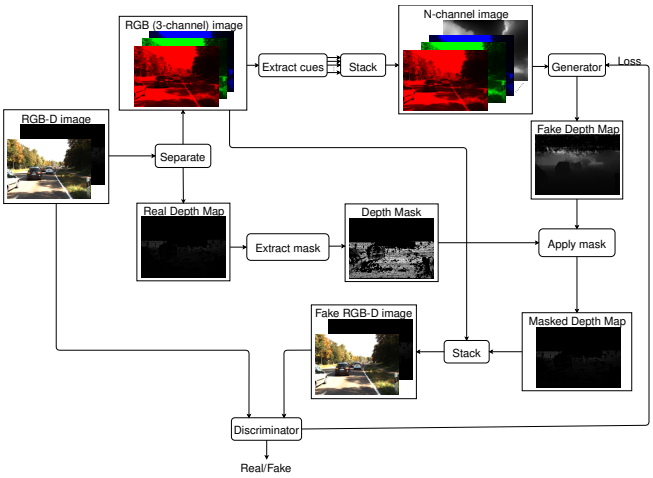

Figure 5. Schematic that shows for a single RGB-D image how it is used to train both the generator and the discriminator of the model.

Al-jibouri changed the number of output channels from 3 to 1 since no RGB image needs to be created but only a depth value for every pixel. For this work, the number of input channels is changed from 3 to N, which allows for the addition of extra cue-related channels to be added to an input image. Since Al-jibouri normalised the ground truth depth data into the interval [1, -1], a hyperbolic tangent [50] was used as the final activation function. For this work, the ground truth depth is not normalised and rectified linear unit (ReLU) [50] is used as the final activation. This allows for absolute values to be produced by the method, which improves the real-world utility and allows for metrics that have a unit to be calculated. Lastly, while Al-jibouri used a fixed image shape of 256x256, this is not required by the model, since it is fully convolutional. Therefore, no fixed image size is used, which does require that at every forward call the output of an upscaling layer is forced into the same shape as the output of its mirror layer. The layers of the generator are described in Table 6.

Before the output of the generator is passed onto the discriminator, two extra transforms are performed. First, the depth values are clamped to the maximum value possible in the ground truth data set (which is almost 256m for Kitti, see Appendix E). Afterwards, the result is masked according to the mask of the ground truth depth map. The maximum possible value and mask are consequences of the way depth data is stored and the limitations of measuring conditions. While they are properties of the ground truth data, they are not inherent properties of depth estimation. Therefore, the two transformations are applied, so that the generator is not incentivised to clamp or mask the output itself. This has two benefits: the generator does not need to suffer from the same limitations as the ground truth and it does not need to allocate capacity for producing features that are not related to depth estimation.

#	Type	Features	Skip	Activation
1	Conv	$N \rightarrow 64$		Leaky
2	Conv	$64 \rightarrow 128$		
3	Norm	128		Leaky
4	Conv	$128 \rightarrow 256$		
5	Norm	256		Leaky
6	Conv	$256 \rightarrow 512$		
7	Norm	512		Leaky
8	Conv	$512 \rightarrow 512$		
9	Norm	512		Leaky
10	Conv	$512 \rightarrow 512$		
11	Norm	512		Leaky
12	Conv	$512 \rightarrow 512$		
13	Norm	512		Leaky
14	Conv	$512 \rightarrow 512$		ReLU
15	Deconv	$512 \rightarrow 512$		
16	Norm	512		
17	Dropout		13	ReLU
18	Deconv	$1024 \rightarrow 512$		
19	Norm	512		
20	Dropout		11	ReLU
21	Deconv	$1024 \rightarrow 512$		
22	Norm	512		
23	Dropout		9	ReLU
24	Deconv	$1024 \rightarrow 512$		
25	Norm	1024	7	ReLU
26	Deconv	$1024 \rightarrow 256$		
27	Norm	256	5	ReLU
28	Deconv	$512 \rightarrow 128$		
29	Norm	128	3	ReLU
30	Deconv	$128 \rightarrow 64$		
31	Norm	64	1	ReLU
32	Deconv	$128 \rightarrow 1$		ReLU

Table 6. Layers of the generator. A description of the terms can be found in Table 8

Discriminator The discriminator has the same configuration as 'pix2pix' [24] and Al-jibouri [2]. The input of the discriminator is a 4-channel image, consisting of the three RGB channels and a channel containing the depth. Similar to the generator, the input is progressively downsampled by the convolutional layers into representations of a smaller shape, but greater depth. The final convolutional layer transforms the deepest representation into a single channel, before the sigmoid activation [50] is applied. All values in this output patch are encouraged to approximate 1 for real RGB-D images and 0 for fake RGB-D images. This is achieved by generating a patch with either only ones or only zeroes during training, depending on the legitimacy

Type	Features	Stride	Activation
Conv	$4 \rightarrow 64$	2	Leaky
Conv	$64 \rightarrow 128$	2	
Norm	128		Leaky
Conv	$128 \rightarrow 256$	2	
Norm	256		Leaky
Conv	$256 \rightarrow 512$	1	
Norm	512		Leaky
Conv	$512 \rightarrow 1$	1	Sigmoid

Table 7. Layers of the discriminator. A description of the terms can be found in Table 8

of the depth map. For Al-jibouri this patch is a fixed shape since the input image has a fixed shape as well. To allow for multiple image dimensions, the desired patch is made the same shape as the output patch for every iteration. This desired patch and the output patch are used to determine the loss, which is propagated back through the layers of the discriminator. The layers of the discriminator can be found in Table 7.

Training During training, RGB-D images are passed through both networks as depicted in Figure 5. While pix2pix uses a batch size of 32, Al-jibouri uses a batch size of 1, this effectively converts the batch normalisation layers into instance normalisation, which has been shown to increase performance for image generation tasks [54]. While this does increase training time, the same batch size is used for this work as there is an additional drawback of using a batch size greater than 1. Namely, images in a batch all need to be of the same size, which is conflict with the goal that training can be done on various image shapes. A solution for this could be to use batches made from single image shapes, but in this process, some of the randomness of the order of training is lost and it is unlikely that all training data can be used every epoch.

During each training iteration, the loss for the generator and discriminator is calculated. As mentioned before, the loss for the discriminator is the binary cross entropy [45] between the output patch and the desired patch. The final loss is the sum of the losses of both the real and the fake RGB-D pair. For the generator, the binary cross entropy loss is calculated for the fake RGB-D pair, but the desired patch is all ones instead of zeroes. Additionally, the L1 loss [58] is calculated between the (masked) fake depth map and the real depth map. The latter is multiplied by 100 before it is added to the former, to ensure both losses are in the same order of magnitude.

Pix2pix uses the Adam optimiser [27] for updating the network weights. For both networks the learning rate is $2 \cdot 10^{-4}$ and β_1 and β_2 are .5 and .999 respectively. Al-jibouri replaced the optimiser for the discriminator with stochastic gradient descent with the same learning rate, as a measure for improving stability. The same optimiser was experimented with for this work, and while stability in the performance during training was improved, the achieved performance was hindered as well. Instead, the Adam optimiser is used again for both networks, but the learning rate is replaced by 10^{-5} and b_1 by .9 for both networks. These measures ensure that a single training instance has less impact on the weights and that the change of each weight is more dependent on previous iterations, respectively. These measures seem to improve stability without noticeably hindering performance.

Type	The kind of layer that is used.
Features	For convolutions the number of input features and output features. For normalisation the number of features.
Skip	The output of which layer is concatenated to the output of the respective layer before the activation function is applied.
Stride	The stride of a convolution or convolutional transpose.
Activation	Activation function that is applied to the output of the respective layer.
Conv	2D-convolutional layer with a certain amount of input and output features, kernel size 4x4 and padding of 1. The stride is 2 for every convolutional layer of the generator, and 1 or 2 for the convolutional layers of the discriminator.
Norm	2D-instance normalisation layer with a certain amount of features, momentum of .1, epsilon of .00001 and no learnable affine parameters.
Deconv	2D-convolutional transpose layer with a certain amount of input and output features, kernel size 4x4, padding of 1, output padding of 0 and stride 2. At every forward call, the output is forced into the same shape as the output of the next skip connection. For the final layer, the output is forced into the same height and width of the input.
Dropout	Layer that randomly makes some of the elements from the output of the previous layer zero with probability .5, as a regularisation measure.
Leaky	Leaky ReLU activation function with .2 as negative slope.
ReLU	ReLU activation function.
Sigmoid	Sigmoid activation function.

Table 8. Descriptions for the terms used in Tables 6 and 7

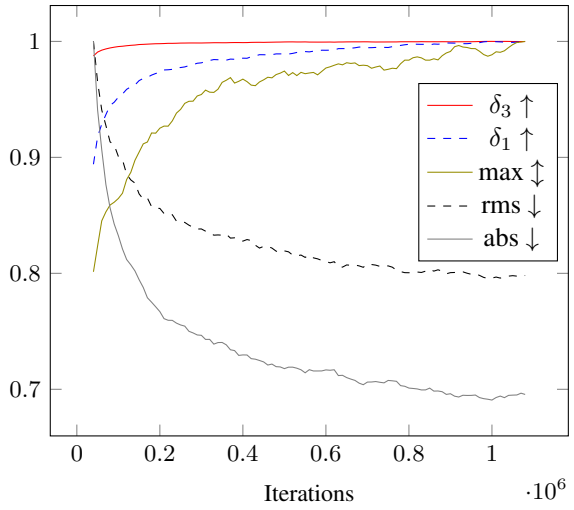

Figure 6. Relative change of different metrics during training of the baseline model (I) for Experiment 1a. These results are achieved by training the model on *Eigen* and evaluating a shapshot of the model on the test set every 10.000 iterations. Each value is determined by calculating the average from a window of 7 snapshots and dividing it by the maximum observed value of the metric.

Appendix B. Maximum Depth

Originally, **max**imum depth was used as a metric. The recorded depth of a data set can be limited by the maximum distance that can be measured, resulting in the ground truth depth map containing many unmeasured pixels. This is prevalent in the Kitti data set, where depth values are not greater than 90 meters (see Appendix E). The output of the generator is masked according to the pixels that have a ground truth value (see Figure 5), so that it is still incentivised to produce depth values for pixels that do not have a ground truth depth value. Consider two depth estimators: one that produces a high-quality complete depth map and one that produces the same quality of depth for the masked pixels but produces arbitrary values for the other pixels. The former would be considered to have a better understanding of the depth, but they would have equal performance for all other metrics. However, the former would have a higher maximum depth.

To determine **max**, the maximum value that is predicted (before any mask is applied) is determined for every test image, after which the average value over all test images is determined. While it cannot be known for certain that greater maximum depth corresponds to a better model, Figure 6 does show that **max** follows a similar path to the ↑-metrics. It has low scores and quick growth early on, and high scores and slow growth later on. Additionally, while the other metrics seem to reach a horizontal asymptote, **max** seems to grow linearly (albeit slowly) after about 400.000 iterations.

IB	IL	IO	IF	IR	$IRFO$
.083	.138	.142	.478	<u>.962</u>	**.965**
.332	.552	.568	.956	**1.05**	<u>1.05</u>

Table 9. The **cue** dependency ration results for Experiment 1b. The first row consists of the original values, the second row of those values normalised by the number of channels in the cue (multiplied by $^N/_{N_c}$, N_c is the number of cue channels). **Bold** represent the greatest **cue** score, while <u>underlined</u> the second.

Furthermore, a qualitative visual investigation lead to the observation that later snapshots predicted depth values for unlabelled areas in images, for which earlier snapshots did not.

All of these observations are supportive of the idea that a greater **max** corresponds to the model having a better 'depth understanding' without simply learning to produce values for the labelled pixels. On the other hand, cue models that significantly outperform the baseline in all the other metrics do not get greater **max** scores. This makes the maximum depth metric less relevant for this work. Additionally, the metric is not compatible with state-of-the-art methods that use bins (AdaBins and BinsFormer), since these use a limited depth range. Although **max** could still correspond to 'depth understanding', any decisive conclusion cannot be drawn. Because of these reasons, the maximum depth metric is removed from the core of this research but kept as an Appendix.

Appendix C. Cue Dependency Ratio

The **cue** dependency ratio was also originally used as a metric. It measures how the weights of the first convolutional layer are linked to the channels of each cue:

$$\frac{1}{|\mathcal{W}|} \sum_{\mathcal{K} \in \mathcal{W}} \left[\frac{1}{\sum_{w \in \mathcal{K}} |w|} \sum_{w \in \mathcal{K}_c} |w| \right]$$

$\mathcal{W}$ contains all the kernels $\mathcal{K}$ of the first convolutional layer of the generator. Each kernel consists of weights w that are not biases, $\mathcal{K}_c \subseteq \mathcal{K}$ contains only the weights in kernel $\mathcal{K}$ corresponding to cue channels. The normalisation of the weights refers to the fact that every value w is the actual weight multiplied by the average value of the image channel it links to. This average is taken from all the non-zero values from that image channel in the training set.

Table 9 shows the results of this metric for Experiment 1a. It immediately becomes clear how the metric is biased towards the number of channels a cue has. A cue not improving depth estimation, does not result in the model making all weights linked to that cue's channels (close to) zero. Normalising the values according to the number of channels generates scores that are somewhat in line with the scores of other metrics. However, IO scores only slightly better than

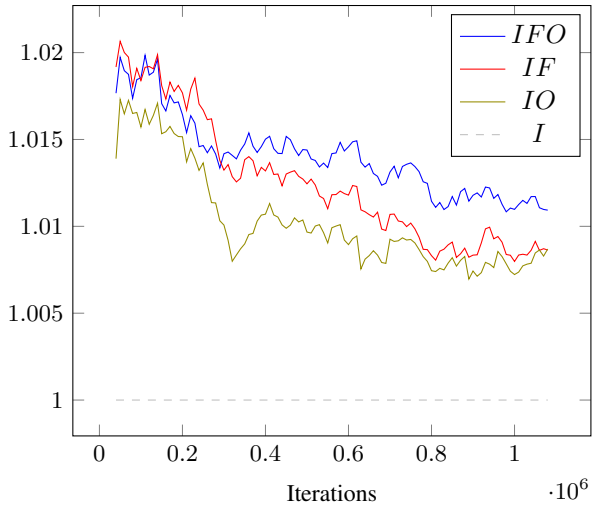

Figure 7. Thresholded difference δ_1 (↑) of different models from Experiment 1a during training. These results are achieved by training the model on *Eigen* and evaluating a shapshot of the model on the test set every 10.000 iterations. Each value is determined by calculating the average from a window of 7 snapshots and dividing it by the 7-average of the baseline model.

IL, and $IRFO$ does not perform better than IR. Because of these unmeaningful results and the fact that normalising the scores does not make the metric a ratio anymore, the cue dependency ratio has been excluded from the core of this research.

Appendix D. IFO

Originally, IFO was included as an extra model in Experiment 1. Familiar size (F) generates the same values for neighbouring objects, which could be misinterpreted by an MDE method as them being a single object. Since overlap (O) generates different values for neighbouring objects, adding it to the input could mitigate that problem and improve performance. Since the results of Experiments 1a and 1b did not support this, IFO has been excluded from the core of this research. However, the original training efficiency results (with the model from Subsection 4.1) did show that IFO can improve upon the performance of IF, see Figure 7.

Appendix E. Data sets

Figure 8 shows the differences between a sparse depth map (KITTI) and a dense depth map (the contributed virtual data set). Figure 9 shows images from different virtual data sets.

KITTI The KITTI data set was collected by Geiger et al. using a car fitted with multiple sensors. These sensors included both grey-scale and colour binocular cameras, and a

Figure 8. RGB-depth pair for both KITTI (top, sparse depth) and the virtual data set (bottom, dense depth).

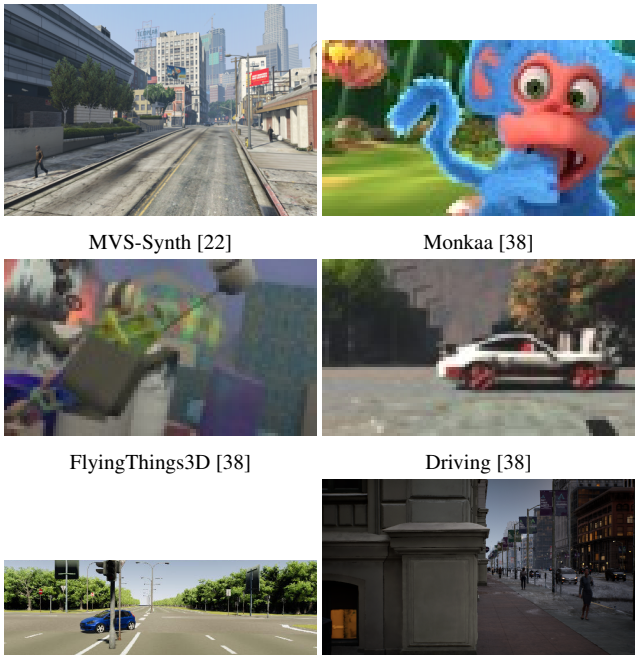

Figure 9. RGB images present in different virtual data sets.

rotating 3D laser scanner. The car was driven along German roads. The recorded data was processed into a data set that includes binocular images with ground-truth sparse depth data for either image. The data is split into train and test sets containing around 57 and 23 thousand images respectively [16]. The depth data is stored as 16-bit unsigned integer values, which have a maximum possible value of 65535. Dividing these values by 256 converts them to meters, which makes the greatest possible distance that can be stored a bit less than 256 meters. Figure 10 shows that the greatest distance in the KITTI training set is 90 meters, which is well within range.

References

[1] Mehrnaz Farokhnejad Afshar, Zahra Shirmohammadi, Seyyed Amir Ali Ghafourian Ghahramani, Azadeh Noorpar-

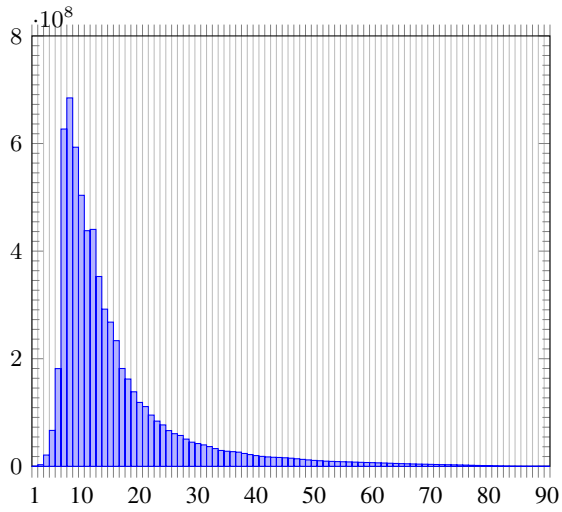

Figure 10. Histogram of non-zero distances (rounded to meters) of pixels in the training set of the KITTI data set [16]. Pixels with values higher than 90 meters do not exist in this set.

var, and Ali Mohammad Afshin Hemmatyar. An Efficient Approach to Monocular Depth Estimation for Autonomous Vehicle Perception Systems. *Sustainability*, 15(11), 2023. ISSN 2071-1050. doi: 10.3390/su15118897. URL https://www.mdpi.com/2071-1050/15/11/8897.

[2] Zina Al-jibouri. Improving depth estimation in an automated privacy-preserving video processing system. Master's thesis, Radboud University, Nijmegen, NL, April 2020.

[3] D. Auty and K. Mikolajczyk. Monocular Depth Estimation Using Cues Inspired by Biological Vision Systems. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 4051–4057, Los Alamitos, CA, USA, aug 2022. IEEE Computer Society. doi: 10.1109/ICPR56361.2022.9956454. URL https://doi.ieeecomputersociety.org/10.1109/ICPR56361.2022.9956454.

[4] Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14:115–133, 1994.

[5] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. AdaBins: Depth Estimation Using Adaptive Bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4009–4018, June 2021.

[6] Yohann Cabon, Naila Murray, and M. Humenberger. Virtual KITTI 2. *ArXiv*, abs/2001.10773, 2020. URL https://api.semanticscholar.org/CorpusID:210942959.

[7] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. doi: 10.1109/TPAMI.1986.4767851.

[8] Ingrid Carlbom and Joseph Paciorek. Planar geometric projections and viewing transformations. *ACM Computing Surveys (CSUR)*, 10(4):465–502, 1978.

[9] Guang Chen, Hu Cao, Jorg Conradt, Huajin Tang, Florian Rohrbein, and Alois Knoll. Event-Based Neuromorphic Vision for Autonomous Driving: A Paradigm Shift for Bio-Inspired Visual Sensing and Perception. *IEEE Signal Processing Magazine*, 37(4):34–49, 2020. doi: 10.1109/MSP.2020.2985815.

[10] Richard J. Chen, Faisal Mahmood, Alan Loddon Yuille, and N. Durr. Rethinking Monocular Depth Estimation with Adversarial Training. *ArXiv*, abs/1808.07528, 2018. URL https://api.semanticscholar.org/CorpusID:52076600.

[11] Christian Dornhege Wolfram Burgard Christian Zimmermann, Tim Welschehold and Thomas Brox. 3D Human Pose Estimation in RGBD Images for Robotic Task Learning. In *IEEE International Conference on Robotics and Automation, ICRA*, 2018. URL https://lmb.informatik.uni-freiburg.de/projects/rgbd-pose3d/.

[12] Lucía Conde Moreno. Automated Privacy-Preserving Video Processing through Anonymized 3D Scene Reconstruction. Master's thesis, Utrecht University, Utrecht, NL, September 2019.

[13] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. doi: 10.1109/iccv.2015.316. URL http://dx.doi.org/10.1109/ICCV.2015.316.

[14] David Eigen, Christian Puhrsch, and Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *NIPS*, 2014. URL https://api.semanticscholar.org/CorpusID:2255738.

[15] Itzhak Fogel and Dov Sagi. Gabor filters as texture discriminator. *Biological cybernetics*, 61(2):103–113, 1989.

[16] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[17] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.699. URL http://dx.doi.org/10.1109/CVPR.2017.699.

[18] S. Golestaneh and Lina Karam. Spatially-Varying Blur Detection Based on Multiscale Fused and Sorted Transform Coefficients of Gradient Magnitudes. 03 2017.

[19] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T. Freeman. Unsupervised Semantic Segmentation by Distilling Feature Correspondences. *ArXiv*, abs/2203.08414, 2022. URL https://api.semanticscholar.org/CorpusID:247476291.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. doi: 10.1109/cvpr.2016.90. URL http://dx.doi.org/10.1109/cvpr.2016.90.

[21] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. doi: 10.1109/iccv.2017.322. URL http://dx.doi.org/10.1109/ICCV.2017.322.

[22] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning Multi-View Stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] J. Illingworth and J. Kittler. A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116, 1988. ISSN 0734-189X. doi: https://doi.org/10.1016/S0734-189X(88)80033-1. URL https://www.sciencedirect.com/science/article/pii/S0734189X88800331.

[24] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.632. URL http://dx.doi.org/10.1109/CVPR.2017.632.

[25] Iliya V. Ivanov, Daniel J. Kramer, and Kathy T. Mullen. The role of the foreshortening cue in the perception of 3D object slant. *Vision Research*, 94:41–50, 2014. ISSN 0042-6989. doi: https://doi.org/10.1016/j.visres.2013.10.019. URL https://www.sciencedirect.com/science/article/pii/S0042698913002617.

[26] Kevin Karsch, Ce Liu, and Sing Bing Kang. Depth Transfer: Depth Extraction from Video Using Non-Parametric Sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2144–2158, nov 2014. doi: 10.1109/tpami.2014.2316835. URL https://doi.org/10.1109%2Ftpami.2014.2316835.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[29] Aditya Kulkarni, Tharun Mohandoss, Daniel Northrup, Ernest Mwebaze, and Hamed Alemohammad. Semantic Segmentation of Medium-Resolution Satellite Imagery using Conditional Generative Adversarial Networks. *ArXiv*, abs/2012.03093, 2020. URL https://api.semanticscholar.org/CorpusID:227335388.

[30] Lubor Ladický, Jianbo Shi, and Marc Pollefeys. Pulling Things out of Perspective. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–96, 2014. doi: 10.1109/CVPR.2014.19.

[31] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. *2016 Fourth International Conference on 3D Vision (3DV)*, Oct 2016. doi: 10.1109/3dv.2016.32. URL http://dx.doi.org/10.1109/3DV.2016.32.

[32] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation. *ArXiv*, abs/1907.10326, 2019. URL https://api.semanticscholar.org/CorpusID:198229801.

[33] Zhenyu Li, Xuyang Wang, Xianming Liu, and Junjun Jiang. BinsFormer: Revisiting Adaptive Bins for Monocular Depth Estimation. *ArXiv*, abs/2204.00987, 2022. URL https://api.semanticscholar.org/CorpusID:247939833.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science*, page 740–755, 2014. ISSN 1611-3349. doi: 10.1007/978-3-319-10602-1_48. URL http://dx.doi.org/10.1007/978-3-319-10602-1_48.

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021. doi: 10.1109/ICCV48922.2021.00986.

[36] Mostafa Mansour, Pavel Davidson, Oleg Stepanov, and Robert Piché. Relative Importance of Binocular Disparity and Motion Parallax for Depth Estimation: A Computer Vision Approach. *Remote Sensing*, 11(17), 2019. ISSN 2072-4292. doi: 10.3390/rs11171990. URL https://www.mdpi.com/2072-4292/11/17/1990.

[37] George Mather and David R R Smith. Blur Discrimination and its Relation to Blur-Mediated Depth Perception. *Perception*, 31(10):1211–1219, 2002. doi: 10.1068/p3254.

URL https://doi.org/10.1068/p3254. PMID: 12430948.

[38] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[39] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52:99–115, 1990.

[40] Sidharth Mishra, Uttam Sarkar, Subhash Taraphder, Sanjoy Datta, Devi Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram. Principal Component Analysis. *International Journal of Livestock Research*, page 1, 01 2017. doi: 10.5455/ijlr.20170415115235.

[41] Takanori Okoshi. *Three-dimensional imaging techniques*. Elsevier, 2012.

[42] Gian F Poggio and Tomaso Poggio. The analysis of stereopsis. *Annual review of neuroscience*, 7(1):379–412, 1984.

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, page 234–241, 2015. ISSN 1611-3349. doi: 10.1007/978-3-319-24574-4_28. URL http://dx.doi.org/10.1007/978-3-319-24574-4_28.

[45] Usha Ruby and Vamsidhar Yendapalli. Binary cross entropy with deep learning technique for Image classification. *International Journal of Advanced Trends in Computer Science and Engineering*, 9, 10 2020. doi: 10.30534/ijatcse/2020/175942020.

[46] Pablo Revuelta Sanz, Belén Ruiz Mezcua, and José M Sánchez Pena. Depth estimation-an introduction. In *Current Advancements in Stereo Vision*. IntechOpen, 2012.

[47] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3D scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2008.

[48] Andre M. Schreiber, Minsik Hong, and Jerzy W. Rozenblit. Monocular Depth Estimation using Synthetic Data for an Augmented Reality Training System in Laparoscopic Surgery. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2121–2126, 2021. doi: 10.1109/SMC52423.2021.9658708.

[49] Shuwei Shao, Zhongcai Pei, Weihai Chen, Ran Li, Zhong Liu, and Zhengguo Li. URCDC-Depth: Uncertainty Rectified Cross-Distillation with CutFlip for Monocular Depth Estimation. *ArXiv*, abs/2302.08149, 2023. URL https://api.semanticscholar.org/CorpusID:256900951.

[50] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6 (12):310–316, 2017.

[51] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[52] Nicola Strisciuglio, Manuel Lopez-Antequera, and Nicolai Petkov. Enhanced robustness of convolutional networks with a push–pull inhibition layer. *Neural Computing and Applications*, 32(24):17957–17971, 2020.

[53] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[54] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *ArXiv*, abs/1607.08022, 2016. URL https://api.semanticscholar.org/CorpusID:16516553.

[55] Zhenda Xie, Zigang Geng, Jingcheng Hu, Zheng Zhang, Han Hu, and Yue Cao. Revealing the Dark Secrets of Masked Image Modeling. *ArXiv*, abs/2205.13543, 2022. URL https://api.semanticscholar.org/CorpusID:249097401.

[56] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Simon Chen, Yifan Liu, and Chunhua Shen. Towards Accurate Reconstruction of 3D Scene Shape from A Single Monocular Image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–21, 2022. ISSN 1939-3539. doi: 10.1109/tpami.2022.3209968. URL http://dx.doi.org/10.1109/TPAMI.2022.3209968.

[57] Wei Zhang, Guoying Zhang, and Qiran Zou. Depth Prediction from Monocular Images with CGAN. In *International Conference on Smart Computing and Communication*, pages 427–436. Springer, 2018.

[58] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017. doi: 10.1109/TCI.2016.2644865.

[59] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 Million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464, 2018. doi: 10.1109/TPAMI.2017.2723009.

[60] Manyu Zhu, Dongliang He, Xin Li, Chao Li, Fu Li, Xiao Liu, Errui Ding, and Zhaoxiang Zhang. Image Inpainting by End-to-End Cascaded Refinement With Mask Awareness. *IEEE Transactions on Image Processing*, 30:4855–4866, 2021. doi: 10.1109/tip.2021.3076310. URL https://doi.org/10.1109%2Ftip.2021.3076310.

[61] Jure Žbontar and Yann LeCun. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. 2015. doi: 10.48550/ARXIV.1510.05970. URL https://arxiv.org/abs/1510.05970.