Combined Depth and Semantic Segmentation from Synthetic Data and a W-Net Architecture

Kevin Swingler[1][a], Teri Rumble[1], Ross Goutcher[2][b], Paul Hibbard[2][c], Mark Donoghue[2] and Dan Harvey[3]
[1]*Computing Science and Mathematics, University of Stirling, Stirling, FK9 4LA, U.K.*
[2]*Psychology, University of Stirling, Stirling, FK9 4LA, U.K.*
[3]*Neuron5, Scotland, U.K.*
{*kevin.swingler, ross.goutcher, paul.hibbard, mark.donoghue*}*@stir.ac.uk, dan@neuron5.co.uk*

Keywords: Computer Vision, Semantic Segmentation, Monocular Depth Estimation, Synthetic Data.

Abstract: Monocular pixel level depth estimation requires an algorithm to label every pixel in an image with its estimated distance from the camera. The task is more challenging than binocular depth estimation, where two cameras fixed a small distance apart are used. Algorithms that combine depth estimation with pixel level semantic segmentation show improved performance but present the practical challenge of requiring a dataset that is annotated at pixel level with both class labels and depth values. This paper presents a new convolutional neural network architecture capable of simultaneous monocular depth estimation and semantic segmentation and shows how synthetic data generated using computer games technology can be used to train such models. The algorithm performs at over 98% accuracy on the segmentation task and 88% on the depth estimation task.

1 INTRODUCTION

Pixel level scene understanding requires an algorithm to label each individual pixel in an image with information regarding its role in the scene. Semantic segmentation involves labelling each pixel as belonging to one of a set of pre-defined classes and panoptic segmentation requires the algorithm to separate different instances of each class. Depth estimation involves labelling each pixel with an estimated distance from the camera. Depth estimation can act either on stereo images, in which pairs of images are taken from specialist devices with two cameras fixed a small distance apart or on single monocular images. Monocular depth estimation is more difficult than the stereo task as there is no image disparity information available. In the particular situation where video data is taken from a moving object, it is possible to estimate depth from changes from one frame to the next.

We have previously shown (Goutcher et al., 2021) that adding semantic segmentation information to training data improves the quality of the depth estimation predictions in the stereo depth estimation task but this introduces the need for training data that have been labelled at a pixel level with both depth and semantic segmentation tags. Labelling depth ground truth may be performed using specialist equipment such as LiDAR, but semantic segmentation labelling is still a time intensive human activity and prone to error.

Software for generating life-like scenes in computer games such as Unity and Unreal is known as a *games engine*. The quality of the graphics produced by these engines has increased dramatically in recent years and the images are now sufficiently life-like that they can be used for training computer vision algorithms. The use of games engines allows us to generate large quantities of quality training images that are automatically and correctly labelled with both depth and semantic category information at the pixel level. This removes the need for human labelling by hand, meaning the data are error free. The human work of labelling is replaced by the human work of programming the games engine, but the investment in time has a better return as it is easy to change the parameters of an artificial environment and produce a new dataset, whereas re-collecting and labelling real world data is very time consuming. For example, we are able to generate the same dataset under a variety of condi-

[a] https://orcid.org/0000-0002-4517-9433
[b] https://orcid.org/0000-0002-0471-8373
[c] https://orcid.org/0000-0001-6133-6729

Swingler, K., Rumble, T., Goutcher, R., Hibbard, P., Donoghue, M. and Harvey, D.
Combined Depth and Semantic Segmentation from Synthetic Data and a W-Net Architecture.
DOI: 10.5220/0012877500003837
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Joint Conference on Computational Intelligence (IJCCI 2024)*, pages 413-422
ISBN: 978-989-758-721-4; ISSN: 2184-3236
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

tions by varying lighting, weather, time of year and time of day.

It is possible to produce both monocular and binocular datasets using a games engine. Training data can be generated as a set of snapshots from random locations in an environment or as a series of video frames constructed by moving a camera through the environment. Cameras may move in a simple pattern such as following a straight line or rotating around a point, or they can be virtually mounted to a vehicle that is programmed to follow the roads in the environment. This method produces more realistic sequences for training models to operate in similar environments in the real world. The games engine can tag each pixel in an image with semantic or panoptic class labels and with a depth (distance from camera) measurement.

The paper makes the following contributions: We describe a new architecture capable of predicting depth and semantic class labels from monocular images. The architecture is based on the well known U-Net (Ronneberger et al., 2015) model, but has two connected U-Nets, leading us to call it a W-Net architecture. We compare different cost functions and describe an approach to depth estimation that treats depth as a category rather than a continuous variable. The paper also describes the benefits of training machine vision algorithms on data generated by a games engine.

2 BACKGROUND

2.1 Depth Estimaton

Active detection systems such as LIDAR (Park et al., 2018) and RADAR (Long et al., 2021) can be extremely precise and helpful in solving the depth estimation problem but are not suitable for mass deployment due to cost and processing and power requirements. Passive systems utilising infrared light such as RGBD cameras are cheaper than both LIDAR and RADAR. They calculate depth information accurately through measuring the dispersal of a known infrared light pattern within the scene, similar to LiDAR. However, they do have some limitations as reported by Alhwarin et al. (Alhwarin et al., 2014). They found that RGBD cameras struggled to deal with both highly reflective surfaces and light absorbing materials. Furthermore, they noted that the use of multiple RGBD cameras in the same scene resulted in the IR patterns interfering with each other, as can happen with LiDAR if the frequencies match. These constraints would be somewhat problematic under conditions of mass deployment without adding the layers of complexity that the authors suggest in order to mitigate against these issues.

Stereo vision depth research focuses on techniques inspired by human vision including triangulation through parallel or intersecting optical axis binocular vision (Wang et al., 2017) and estimation of depth through binocular disparity cues (Mansour et al., 2019). Whilst very effective these are not very practical for affordable mass deployment due to requiring two cameras (therefore twice the cost as well as ongoing maintenance to ensure precise calibration) which inevitably constrains the robustness of these solutions in an applied setting. It is clear that accurate monocular depth estimation would be preferable on this basis. Monocular depth perception has taken great strides in recent years due to the ability of CNNs to learn complex monocular depth cue patterns such as object size, texture, and linear perspective. However, it is not without its own set of problems, such as resolving the focal length of the camera and image defocus and blur as summarised by Paul et al. (Paul et al., 2022). A novel approach to extracting more depth information from a single image was demonstrated by Lee et al. (Lee et al., 2013) using dual aperture cameras to simultaneously capture red and cyan filtered light in an attempt to establish a depth relationship between the information contained in the differing light channels. Although they successfully demonstrated that this method is viable as a means of estimating depth from a single monocular source, it does require the use of specialised camera equipment.

There are several monocular depth estimation techniques that use temporal image sequencing (Palou and Salembier, 2012), (Li et al., 2021), (Chen et al., 2020) instead of binocular images. Using sequential frames from a video feed allows researchers to simulate binocular left and right image pairs if the monocular frame rate and speed of motion is carefully considered. Depth estimation from motion has the advantages of a monocular approach in that it only requires a single commodity camera but introduces a number of technical challenges if the speed of motion varies. It is easy to produce images sequences from a games engine with a constant inter-frame off-set but this is not a constraint that is practical in real world applications. A simple monocular approach has the additional benefit of requiring fewer processing resources than either a stereo or a temporal off-set approach as it acts on one frame instead of two.

Perhaps the most well known monocular depth estimation approach is an algorithm known as MiDaS (Ranftl et al., 2020). The authors note that a lack of diverse training data is a limiting factor for building depth estimation models and propose a system capable of combining data from different sources. As we have already noted, collecting depth information for training networks is more costly and technically challenging that the task of collecting images for tasks such as classification or object detection as many standard sources of image data lack a depth channel.

The depth estimation problem is almost exclusively treated as a regression task in the literature. Notable exceptions are Cao et al. (Cao et al., 2017) and Su et al. (Su et al., 2019), who both explore the possibility of casting depth estimation as a classification task. Depth may be quantized into a set of bins with the depth estimate being taken as the bin with the highest likelihood. This method provides a natural representation of confidence levels for predictions as more than one possible depth can be represented at once. We test the hypothesis that treating the depth estimate as a classification task allows for sharper changes at the edges of objects.

2.2 Image Segmentation

Segmentation tasks broadly fall into one of three categories. Semantic segmentation (Liu et al., 2019) attempts to label each pixel in an image by its class. Some classes are objects (such as cars and people) and other classes are known simply as *stuff*, which includes sky, grass, foliage, etc. Instance segmentation (Tian et al., 2021) aims to label individual objects (car1, car2 etc.) but ignores pixels that belong to the *stuff* class. Panoptic segmentation (Kirillov et al., 2019) labels all pixels (objects and stuff) but separates objects into instances.

Deep learning, specifically the fully convolution network (FCN) (Long et al., 2015), has proven to be very useful in the task of scene understanding through segmentation. Ronneberger et al. (Ronneberger et al., 2015) introduced U-Net as the first encoder-decoder network for semantic segmentation along with the skip connection. The U-Net architecture and its variants remains one of the most effective encoder-decoder networks in the field of semantic segmentation. U-Net++ was proposed by Zhou et al. (Zhou et al., 2019) in 2020, followed by TMD U-Net (Tran et al., 2021) in 2021 by Tran et al. to overcome one of the main limitations of U-Net in that the skip connection requires feature fusion at matching scales. Whilst not focusing on skip connection limitations, our own research builds upon the basic U-Net and demonstrates that it is an architecture capable of being utilised within the domain of simultaneous segmentation and depth estimation in the style of multi-task learning.

More advanced algorithms such as STEGO (Self-supervised Transformer with Energy based Graph Optimisation) (Hamilton et al., 2022) and DINO (Caron et al., 2021) have also been used for semantic segmentation. STEGO combines the two key innovations from DINO, namely transformers and self-supervised learning in the form of self-distillation, with contrastive clustering. Transformers may have gained in popularity in part due to their efficiency in requiring substantially less computational resources but this comes at a price. A drawback of the underlying vision transformer (ViT) architecture is that ViTs require a huge amount of data (Dosovitskiy et al., 2020) to achieve parity with, or to beat state of the art CNNs (approx. 14m+ images) which makes them unfeasible for smaller projects with niche datasets when training from scratch. U-Net on the other hand performs extremely well even with smaller datasets.

Panoptic segmentation is relatively new. Proposed by Kirillov et al. (Kirillov et al., 2019), panoptic segmentation combines semantic and instance segmentation into the single task of labelling each individual instance of each class separately. In this sense, it is very similar to the well-known object detection task, where individual objects are identified and located within a rectangular bounding box. The introduction of pixel level depth estimation makes the panoptic segmentation task easier as pixels that are adjacent in the flat image plane can be separated in the three dimensional representation that follows depth segmentation.

3 DATA SETS

One of the biggest challenges to investigating the full potential of deep learning networks is the availability of high-quality datasets with suitable annotations. Pixel level annotations of both depth and semantic segmentation can be particularly expensive to produce. To address this issue, we developed a synthetic dataset using the Unity games engine, giving us full control over the scene contents and variables such as weather, visibility and time of day, in order to produce a suite of progressively more difficult scenes on which to train and refine a model. This vastly sped up the research cycle and facilitated fast, iterative development without incurring excessive time or financial costs to the research schedule.

Procedural scene generation uses algorithms to produce landscapes, vegetation, varied terrain, sky conditions (like clouds) and bodies of water. Roads and streets that follow the terrain can be added and buildings, vehicles, other man-made objects, and people can be placed in a stochastic fashion. This ensures both great variation in scene content and a realistic arrangement of objects.

Images are generated from the scene from virtual cameras that can be fixed or move along a predetermined path. Once a world has been defined, the following data collection tasks are made easy:

- Altering the number of examples of objects from each class to control class bias
- Altering weather conditions, time of day, lighting, and season (for example rendering trees with or without leaves)
- Changing the camera angle and viewpoint
- Automatically labelling every pixel in the image with depth and semantic labels.

3.1 Custom Dataset Creation

The dataset used in this study was produced with digital content creation (DCC) software SideFX's Houdini version 19.0.498 and the real-time rendering engine Unity version 2021.1.28f with some pre-made asset packs available on the Unity store. Houdini is frequently used to produce environments for films and games because it allows users to build complex procedural content without having to write custom algorithms for each use case. Unity is most often used as a games engine but recently has increasingly been used in computer vision applications because of its ability to render large volumes of digital content. Here Unity is supported by the open-source package Unity Perception (version 0.9.0 preview 2) that adds common functionality for computer vision applications. Houdini has excellent integration with Unity through the Houdini Engine for Unity (version.4.2.8) that allows Houdini node graphs to be packed into a Houdini Digital Asset (HDA) that can then be rendered in Unity. Procedural content can be developed in Houdini, rendered in Unity and automatically annotated with the relevant pixel level ground truths such as depth and segmentation label.

Scenes were rendered as if taken from cameras fixed to a vehicle driving along a road that winds through the virtual world. Three virtual cameras were used, one looking forward, and one each looking left and right. Terrain, object placement and lighting were randomly varied. Object placement utilised either a physics-based ground collision system (more suitable for loose objects), or a ray casting system (more suitable for objects that are fixed to the ground surface), such that objects settled in natural positions consistent with the surface terrain. An example of a training

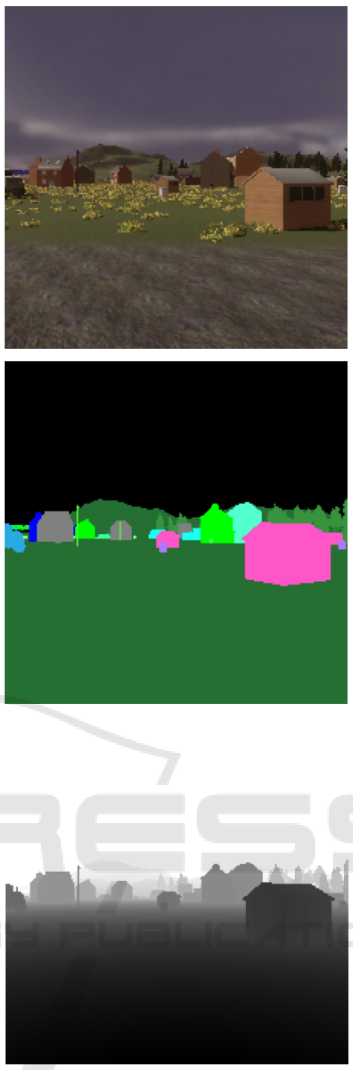
Figure 1: An example image and associated data, showing the natural scene, the semantic segmentation and the depth map as generated by the Unity games engine. In the depth map, the lighter the grayscale, the further from the camera the pixel lies.

image, segmentation map and depth map are shown in figure 1.

The world from which images were sampled was an area of trees and hills dotted with settlements of different buildings, vehicles and man-made objects. A two stage hierarchy of class labels was used. Firstly, pixels were identified as being either *objects of interest* or as *surroundings* such as ground, sky or foliage. Each pixel was also labelled as belonging to a single class such as house, car, truck, etc. There were 19 different classes in the data. The class labels were used to train the network and the differentiation between objects and surroundings was used to calculate performance metrics on both the scene as a whole and on objects of interest alone.

3.2 Scene Statistics of a Custom Dataset

Our synthetic scenes were validated by statistical comparison with natural scenes. Images of the natural world have predictable statistical properties, reflecting structural constraints such as the presence of the ground plane and vertical structures such as trees and buildings (Simoncelli, 2003), (Hibbard and Bouzit, 2005), (Hibbard, 2007). We verify the validity of our dataset by demonstrating that our scenes have the same statistical properties as natural scenes. The critical statistic in this case is the distribution of distances to visible points. Analysis of LIDAR images has shown that the probability distribution of distances to visible points falls exponentially with distance (Yang and Purves, 2003). This distribution appears as a straight line when plotted on log-log axes. The probability distribution of distance for our scenes, plotted in figure 2, is close to the expected exponential distribution.

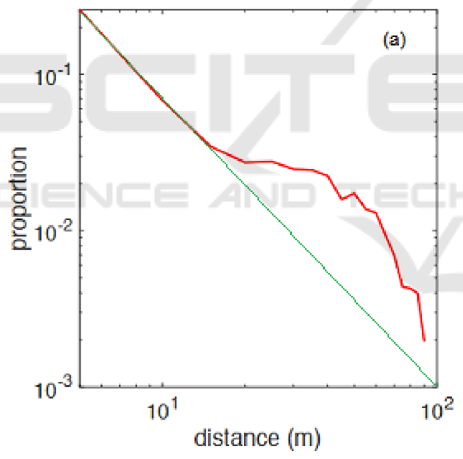

Figure 2: The distribution of distances in the images shows the expected exponential distribution (shown on a log-log scale). The red line shows the distribution of distances in the synthetic data and the green line shows a theoretical natural relationship.

4 ARCHITECTURE AND TRAINING

The W-Net architecture introduces two novel approaches. The first novel approach is the use of back-to-back U-Nets with the first U-Net performing semantic segmentation and the second U-Net performing depth estimation. The second novel approach is to treat depth estimation as a classification task rather than the typical regression estimation.

4.1 Proposed Architecture

4.1.1 General Network Shape

Figure 3 shows a schematic of the proposed W-Net model architecture. The general shape is of two U-Nets in series, arranged so that the first U-Net receives an image as input and produces a segmentation mask as output. The second U-Net produces the depth map as its output and receives its input from the first U-Net in a way that is explained in more detail below.

The first U-Net has two stages: the encoder stage and the decoder stage. The encoder stage contains a series of four blocks, each with the same set of operations that produce feature sets of different sizes. Each block applies two repetitions of 2D convolution followed by batch normalization. In each block, this doubles the number of channels. Each block then performs 2D max pooling to halve the size of each channel and passes the output from this stage into the next block.

The output from the fourth block passes through one more convolution stage, which we call the bridge, the output of which is used for two different processes. It forms the input to the decoding stage of the segmentation U-Net and it contributes the input to the depth estimation U-Net. The decoder stage of the segmentation U-Net contains a series of four blocks, each of which performs an upsampling operation, then concatenates the channels from its equivalent layer in the encoder stage. The resulting set of filters then pass through a convolution and batch normalisation. The size and shape of each layer changes in a way that increases the size of each channel and reduces the number of channels. The final output layer from the decoder stage produces a tensor of shape $h \times w \times c$ where h is the height of the input image, w is the width of the input image and c is the number of classes the model can classify. The final dimension, c is one-hot encoded in the training data and passed through a softmax function at the output of the model. The class for each pixel is determined as that with the maximal output value.

The depth estimation part (the second U-Net) does not take the original image as its input. Its input is built by a series of upsampling and concatenation operations starting at the bridge layer from the bottom of the first U-Net. A four stage upsampling process takes the bridge layer output and performs an upsample followed by a concatenation with the filters from the encoder channel of the U-Net. No convolutions take place, the filters are simply concatenated and upsampled once for each layer in the U-Net. The process mirrors the decoder channel of the segmentation U-Net in the sense that it upsamples and concatenates the encoder channel in reverse order, but there are no convolutions to reduce the number of channels.

After this input, the depth U-Net has a structure that is almost identical to the segmentation U-Net except that the bridge at the bottom of the U-Net has five layers of convolution and the final output layer encodes the depth output. Two different encodings for depth were compared. In one case, depth was encoded as a single numeric output for each pixel in the image, making depth estimation a regression task. In the other, the task was a classification problem as each pixel used a categorical (one-hot) encoding. The differences between the two approaches are discussed in the results section.

The size of the layers at each point in the network are as follows. Input images are $224 \times 224 \times 3$ and RGB encoded. The segmentation U-Net encoder stage produces layers of size: $112 \times 112 \times 16$, $56 \times 56 \times 32$, $28 \times 28 \times 64$, $14 \times 14 \times 128$ and the segmentation bridge output layer is $14 \times 14 \times 256$. The decoder stage has layers of the same shape, but in reverse order until the output layer of shape $112 \times 112 \times c$ where c is the number of classes.

The depth estimation U-Net input is built as follows. The output from the segmentation bridge and each stage of the segmentation encoder channel is upsampled to a size of 224×224 across all channels. The number of channels in each of those layers, as described above, is 256 in the bridge and 16, 32, 64 and 128 respectively in the encoder layers. Concatenating those channels produces an input to the depth estimation U-Net with shape $224 \times 224 \times 496$, which becomes $112 \times 112 \times 16$ in the next layer. From there, the pattern is the same as that for the segmentation U-Net in both directions (encoder and decoder). The depth decoder output shape is $224 \times 224 \times 1$ when a numeric estimation of depth is given and $224 \times 224 \times 256$ for depth classification, allowing 256 distinct depth values. All of the convolution filters are 3 by 3 with padding to create an output of the same size as the input.

4.2 Hyper Parameters for Training Stability

Weights were initialised using HeNormal (He et al., 2015) initialisation, which is better suited to the use of ReLU activation functions. This was found to lead to more stable training profiles. A bias initialiser

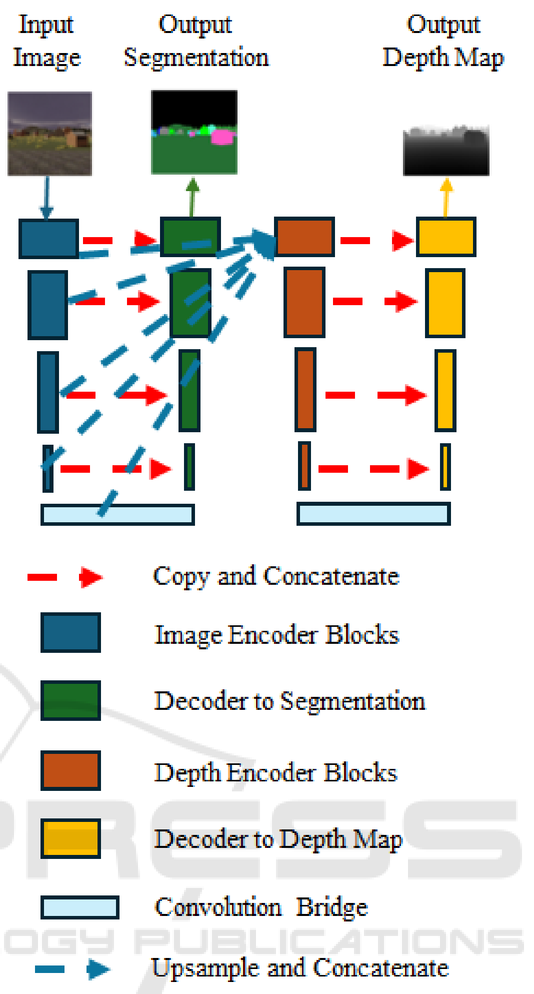

Figure 3: The W-Net architecture consists of two U-Nets. The first U-Net accepts an image as input and produces a segmentation map as output. The input to the second U-Net (the depth predictor) consists of the result of upsampling the segmentation bridge output and each segmentation encoder level, each to a size of 224×224 (the input image size) and then concatenating those channels to form an input shape of $224 \times 224 \times 496$.

term was added (Goodfellow et al., 2016), which further improved stability. It was discovered that setting the bias initialisation to 0.1 produced better results than setting it to 1.0. Dropout was found to be the most effective form of regularisation and 2D Spatial Dropout was added to the encoder layers in increasing amounts (0.1, 0.1, 0.2, 0.3) and applied after the max pooling layer. Applying the dropout in this way ensured the integrity of the skip connection whose job is to provide specific spatial information to the decoder and did not cause any unnecessary fluctuations in the max pooling operations. Dropping entire banks of filters once layer operations are complete forces each level to learn more robustly as seen with dropout in fully connected networks.

The final enhancement for training stability was to switch from the ReLU activation function to eLU. Proposed by Clevert et al. (Clevert et al., 2015), the use of eLU leads to faster and more accurate training. The presence of negative activation values reduces the amount of variance pushed through the forward propagation phase by shifting the mean unit activation gradient closer to zero leading to improved generalisation and accuracy. According to the authors, this activation is particularly helpful to deep networks containing more than 5 layers.

4.3 Output Layer and Loss Function

The network described in this paper uses a categorical output layer for depth estimation, rather than the more usual choice of a single numeric measure of depth. This has the disadvantages of adding a little to model complexity and of limiting the resolution of the depth estimates. It offers the advantage of allowing sharper and larger changes in depth between neighbouring pixels as there is no continuum from one depth class to the next. This is a general difference between regression models, which learn a smooth output function, and classification models, which learn sharp boundaries. We use 256 discrete distance classes, all but one representing a depth bound of one metre. The final class represents a depth of further than 255 metres (in our images, that is mostly the sky).

A series of experiments were performed to compare a numeric depth prediction (i.e. a regression task) with different cost functions against the categorical model. Three different loss functions were compared: Mean Squared Error (MSE) and weighted Structural Similarity Index (SSIM) (Yang and Purves, 2003) for the regression task and categorical cross-entropy for the depth classification task.

Figure 4 shows an example set of results on a single image. The classification model was better able to identify very far objects and was also better at separating the depth of neighbouring pixels that were at very different depths. This confirms the hypothesis that a classification model would be able to learn sharper depth boundaries, compared to a regression model that learns a smooth function.

4.4 Training

Ten percent of the dataset were randomly partitioned as the test set. The remaining data was split randomly with an 80/20 ratio to form the training and validation subsets. The models were trained for 200 epochs with early stopping (patience: 40) and reduced learning rate on plateau conditions (patience: 10, factor: 0.5). The Adam optimiser was used with a learning rate of 0.001 and a constant categorical cross entropy loss function for segmentation, along with a final batch size of 4. It was noticed on the first training run that the decreased batch size had a regulating effect most likely due to the increased noise between training batches and led to a slight increase in accuracy without any other hyper-parameters being changed. The use of decreased learning on a plateau was preferred over a learning rate scheduler as the model had shown the ability to consistently train deep into the epoch cycle.

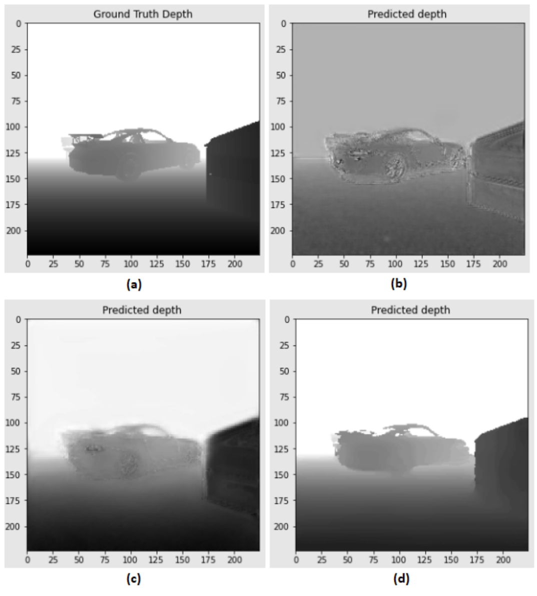

Figure 4: Early loss function experiments showing the quality of depth estimation with three different cost functions: (a) Ground Truth (b) MAE (c) Weighted SSIM (d) classification with Categorical Cross-entropy.

Hyper-parameter	Value
Epochs	200
Optimiser	Adam
Learning Rate	0.0010
Batch Size	4
Early Stopping	40
Reduce LR on Plateau: Patience	10
Reduce LR on Plateau: Factor	0.5

Figure 5: Summary of hyper-parameters used across all experiments & datasets. Reduce Learning Rate (LR) on Plateau allows the network to autonomously adjust its learning rate according to predefined parameters.

5 RESULTS

Segmentation and depth estimation performance results are reported in this section. All results are from images that were taken from the same synthetic environment as the training data, but were not used in the training or hyper-parameter setting. Figure 6 shows example results for a single input image. The target and predicted segmentation maps are shown along with the predicted depth map.

5.1 Segmentation

Across full images, including easier to identify areas such as sky, road and foliage, the model performed at 98.9% accuracy, with a true positive rate of 90% and a false positive rate of 0.1%. We also calculated the accuracy on objects of interest, such as buildings, cars and trucks, but excluding ground, sky, and foliage. This produces a more realistic measure of performance on useful objects. Accuracy was 95% with a true positive rate of 88% and a false positive rate of 0.5%.

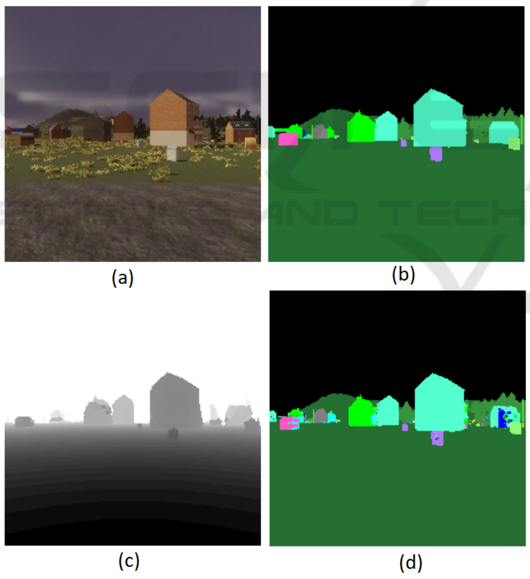

Figure 6: Example outputs from the model: (a) The input image (b) The target segmentation map (c) Model depth estimation (d) Model segmentation output.

5.2 Depth

Depth estimation is treated as a classification task where each category represents a one metre depth zone. We can measure accuracy in terms of the percentage of pixels that are classified into exactly the right depth zone. Across all pixels in the test images, the average correct classification rate was 88.8%. The categories are ordered, however, so we can also measure the size of any error by converting the classes to absolute depth values. Conversion to absolute depth values allows for an understanding of average errors in both absolute terms and proportional to true depth. Overall depth errors were low, with 93% of pixels estimated within a 5% margin of error. This rose to 95% of pixels at a 10% margin of error. For objects of interest (i.e. for pixels with segmentation labels other than ‘terrain’, ‘road’ and other spatially extended items), this value was notably lower at 75% and 88.6% of pixels within 5% and 10% margins of error respectively. These lower values reflect the overall reduction in the number of pixels comprising these objects (i.e. where misestimation of a small number of pixels will have a larger effect on proportional values).

6 CONCLUSIONS AND FUTURE WORK

Previous work has shown that combining depth estimation and segmentation of stereo images improves performance on both tasks and this work builds on that by showing similar results for monocular input images. We have also demonstrated how very high quality training data with precise ground truth can be produced using games technology software. The study is small, with a limited virtual world, a small number of different objects, and a graphics engine that produces images that are clearly artificial. Improvements in graphics realism coupled with the ability to change details such as weather, season, time of day and lighting conditions will improve the data further. We will also need to improve the variation of detail on each object, for example using different road surfaces.

We found that the depth estimation algorithm is able to assign the correct depth to pixels from different instances from the same class, even when one instance partly occludes the other so that their segmentation maps form one continuous shape. This makes the task of panoptic segmentation much easier as instances that appear to be close in the image plane are separated in the depth plane and can be isolated with a simple clustering algorithm such as DBSCAN (Ester et al., 1996). The games engine is able to generate instance level labels, which may be used in future work to develop improved panoptic depth and segmentation algorithms.

As noted by Ranftl et. al. (Ranftl et al., 2020), general purpose depth estimation models require a large quantity of diverse training data. We agree and while there are undoubtedly improvements that could be made to the model architecture and training regime, the single largest opportunity for model improvement will come from generating larger and more diverse training data sets. An important observation from the current work is that the virtual world allows a fine level of control over the arrangement, density and distribution of objects at various depths. For example, we found that early data sets contained too much road and sky, which distorted the accuracy metrics. Later data sets contained more objects of interest and produced more robust models. Future models will be trained on data that is generated in an interactive process, designed to create data for classes and at depths where the network errors are largest. We believe this will allow a greater efficiency for large scale network training.

ACKNOWLEDGEMENTS

This research was funded by DSTL.

REFERENCES

Alhwarin, F., Ferrein, A., and Scholl, I. (2014). Ir stereo kinect: improving depth images by combining structured light with ir stereo. In *Pacific Rim International Conference on Artificial Intelligence*, pages 409–421. Springer.

Cao, Y., Wu, Z., and Shen, C. (2017). Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(11):3174–3182.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660.

Chen, K., Pogue, A., Lopez, B. T., Agha-Mohammadi, A.-A., and Mehta, A. (2020). Unsupervised monocular depth learning with integrated intrinsics and spatio-temporal constraints. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2451–2458. IEEE.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Goodfellow, I., Bengio, Y., Courville, A., and Bach, F., editors (2016). *Deep Learning*, page 193. Adaptive Computation and Machine Learning Series. MIT Press, 12th floor, One Broadway, Cambridge, MA 02142.

Goutcher, R., Barrington, C., Hibbard, P. B., and Graham, B. (2021). Binocular vision supports the development of scene segmentation capabilities: Evidence from a deep learning model. *Journal of vision*, 21(7):13–13.

Hamilton, M., Zhang, Z., Hariharan, B., Snavely, N., and Freeman, W. T. (2022). Unsupervised semantic segmentation by distilling feature correspondences. *arXiv preprint arXiv:2203.08414*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Hibbard, P. B. (2007). A statistical model of binocular disparity. *Visual Cognition*, 15(2):149–165.

Hibbard, P. B. and Bouzit, S. (2005). Stereoscopic correspondence for ambiguous targets is affected by elevation and fixation distance. *Spatial vision*, 18(4):399–411.

Kirillov, A., He, K., Girshick, R., Rother, C., and Dollár, P. (2019). Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413.

Lee, S., Kim, N., Jung, K., Hayes, M. H., and Paik, J. (2013). Single image-based depth estimation using dual off-axis color filtered aperture camera. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2247–2251. IEEE.

Li, S., Luo, Y., Zhu, Y., Zhao, X., Li, Y., and Shan, Y. (2021). Enforcing temporal consistency in video depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1145–1154.

Liu, X., Deng, Z., and Yang, Y. (2019). Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 52(2):1089–1106.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Long, Y., Morris, D., Liu, X., Castro, M., Chakravarty, P., and Narayanan, P. (2021). Radar-camera pixel depth association for depth completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12507–12516.

Mansour, M., Davidson, P., Stepanov, O., and Piché, R. (2019). Relative importance of binocular disparity and motion parallax for depth estimation: a computer vision approach. *Remote Sensing*, 11(17):1990.

Palou, G. and Salembier, P. (2012). 2.1 depth estimation of frames in image sequences using motion occlusions.

In *European Conference on Computer Vision*, pages 516–525. Springer.

Park, K., Kim, S., and Sohn, K. (2018). High-precision depth estimation with the 3d lidar and stereo fusion. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2156–2163. IEEE.

Paul, S., Jhamb, B., Mishra, D., and Kumar, M. S. (2022). Edge loss functions for deep-learning depth-map. *Machine Learning with Applications*, 7:100218.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2020). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Simoncelli, E. P. (2003). Vision and the statistics of the visual environment. *Current opinion in neurobiology*, 13(2):144–149.

Su, W., Zhang, H., Li, J., Yang, W., and Wang, Z. (2019). Monocular depth estimation as regression of classification using piled residual networks. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2161–2169.

Tian, D., Han, Y., Wang, B., Guan, T., Gu, H., and Wei, W. (2021). Review of object instance segmentation based on deep learning. *Journal of Electronic Imaging*, 31(4):041205.

Tran, S.-T., Cheng, C.-H., Nguyen, T.-T., Le, M.-H., and Liu, D.-G. (2021). Tmd-unet: Triple-unet with multi-scale input features and dense skip connection for medical image segmentation. In *Healthcare*, volume 9, page 54. Multidisciplinary Digital Publishing Institute.

Wang, T., Liu, B., Wang, Y., and Chen, Y. (2017). Research situation and development trend of the binocular stereo vision system. In *AIP Conference Proceedings*, volume 1839, page 020220. AIP Publishing LLC.

Yang, Z. and Purves, D. (2003). A statistical explanation of visual space. *Nature neuroscience*, 6(6):632–640.

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., and Liang, J. (2019). Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE transactions on medical imaging*, 39(6):1856–1867.