– Independent Work Report Spring 2019 –

Monocular Depth Refinement with Segmentation Priors and Ordinal Constraints: Towards a Deep Learning Approach

Gabriel Birman
Adviser: Jia Deng

Abstract

This paper considers the addition of prior segmentation masks and user-assisted cues for depth refinement of a base depth estimation network. By training using simulated user-clicks of ordinal relations between the base output and the ground truth depth, the refinement module manages to effectively refine the depth maps of the base network.

1. Introduction

This research acts as a component of a larger project to create 3D representations of objects from 2D images with user assistance. With object segmentation, depth estimation, and shape completion forming the three primary sub-tasks, the goal of this project is to successfully leverage user assistance to accurately estimate depths within the boundaries of a photographed object.

Monocular depth estimation is a well-studied problem in the field of computer vision, with state-of-the-art approaches primarily making use of deep learning techniques on RGB-D datasets to estimate depth maps [5, 6, 7, 9]. Unfortunately, these approaches do not sufficiently capture the information in the scene to provide fully accurate depth maps. Additional information can thus be leveraged to improve the outputted depth map.

In the context of this project, this additional information consists of the segmentation mask of the object in question and user-specified ordinal depth constraints by adding these as training channels to a refinement module and learning a depth refinement. The refinement module serves as an adjunct to a state-of-the-art Resnet-encoded base network [7], wherein the output of the base network, the segmentation mask, and the ordinal depth constraints – an indication of whether a given pixel in the

object mask should be closer or farther than the current prediction suggests – serve as input to the refinement module.

Figures 1, 2, 3, 4 show an example of how the refinement network improves the masked base output. This refinement module performs well on the first iteration, but diverges with further iterations. This project's contribution is to demonstrate an all-around procedure for refining a base network with simple user-simulated ordinal constraints, and consider the implications thereof with respect to holistic depth refinement as well as the specific 2D-3D object creation use case.

Figure 1: Ground truth masked depth map

Figure 2: base network masked depth map

Figure 3: Sampled user clicks

Figure 4: Refined Depth

2. Problem Background and Related Work

Besides this project's use case, depth estimation is important for a variety of tasks in robotics, autonomous vehicles, model reconstruction etc. Although more advanced techniques exist for obtaining depth information from the outside world, such as LiDAR or Infrared, these can be prohibitively expensive and/or unavailable. This project ultimately aims to be ported to mobile phone usage where standard RGB images are taken – although the iPhone has recently made

advances in this space with their TrueDepth camera that takes advantage of infrared sensing [1]. In general, RGB depth estimation is still a widely applicable problem.

Deep learning for image processing makes use of very large scale concatenations of convolutional layers, activation layers, and other processes to effectively learn functions over very high-dimensional data such as images [17]. This has resulted in current depth estimation methodologies implementing convolutional neural networks (CNNs) to achieve state-of-the-art results. As mentioned earlier, even state-of-the-art depth estimation networks have much room for improvement, as capturing three-dimensional information from a two-dimensional representation is not a trivial task.

Segmentation information has been used to improve the network as in [11, 2], but these jointly learn semantic information and depth, i.e. concurrently infer both depth and segmentation maps, as opposed to using ground-truth segmentation maps as input to infer depth, which is what this project proposes.

When it comes to user cues for CNN depth estimation, ordinal depth relations are primarily chosen to aid in the process [18, 4, 12]. Ordinal depth relations, as defined here, refers to a pairwise comparison between points in the input. For instance, given a pair of pixels i and j in the RGB image, the user is asked to rank which one is relatively closer (to the camera plane). This is a simple task for human eye [15], as humans can easily discern metric, as opposed to relative depth. [18, 4] use sparse depth annotations in the loss function as opposed to ground truth depths, while [12] uses ordinal relations to refine a base network.

Ultimately, automatic methods, which infer depth by exploiting latent perception cues, do not perform as well as interactive methods, which respond interactively to sparse user cues to predict a dense depth map [3]. [13, 16] use random walks to generate depth maps; [8] uses a superpixel based geodesic distance weighting interpolation; [10] uses the aforementioned ordinal depth relations. [3] uses user-defined scribbles marking closer/farther regions from the camera;

The common element to all of these approaches is that they solve a non-linear optimization problem to minimize their respective energy function, and so these approaches suffer from the

selection of the energy function. Moreover, the user input is a complicated series of squiggles. This makes it difficult to improve the approaches with deep learning methods, as simulating user input already requires an *a priori* scene understanding, and porting the optimization problems to a CNN is not trivial, although doable. On top of this, the lack of open-source implementations makes the process of reproducing and extending quite time-consuming. [12] appends several ADMM modules to a base network, which solve optimization problems defined by ordinal constraints, but these are limited to very few click pairs (on the order of 10 click pairs), and the CNN is only set up for one iteration of refinement.

A few questions thus arise. What is the relationship between click pairs and depth refinement? Is it possible to successively iterate the refinement modules multiple times? Can a CNN be trained to refine the depth map without specifying an optimization problem to solve? How does the addition of a segmentation mask aid in the depth estimation? This paper seeks to answer all of these questions.

3. Approach

This paper began as an extension of the network proposed in [12]. Unfortunately, the code was proprietary, and there was considerable difficulty in replicating the ADMM module architecture for the purpose of extending it despite much effort. Thus, the approach had to be modified quite late in the process, which resulted in limited computation time for the extremely computationally-expensive and lengthy process of training a deep CNN.

With this in mind, the author chose a similar approach of refining the output of a base network using ordinal constraints, as well as the segmentation mask of the object in question. However, the ordinal relations were no longer relations between pairs in the same RGB image, but rather the same indexed pixel in the output of the base network and the ground truth. This RGB-GroundTruth ordinal relation, akin to the scribbles in [3], achieved better results on a validation set than the RGB-pairwise, and is still a very simple task for the user to compute given that they have a correct depth map in mind. Moreover, this approach allows for the ordinal constraint channel to share maintain local information during the convolution process by maintaining the same shape as the

depth map.

4. Implementation

Due to the computationally-intensive nature of training, many of the decisions below were made based on cursory rather than examining the entire search space, which in this case refers to the cartesian product of hyperparameters. Thus, many hyperparameters/training decisions were made. Rationales for all implementation where an explanation is necessary are provided.

4.1. Base Network and Refinement Module Architecture

Hu et al.'s [7] depth estimation network was chosen as a base network. There were two reasons for this decision. It produced state-of-the-art results. There was an existing implementation online that was not buggy. These reasons are quite sufficient for our purposes, as the refinement module should in theory show improvement on the outputted depth map no matter what kind of depth map was produced. Of course, this was not necessarily the case, as both the base network and the refinement module used similar network architectures. This network architecture consisted of an encoder, decoder, multi-feature fusion, and refinement submodules. Figure 5 provides a visual representation of the underlying feature transformations. This multi-scale architecture preserves spatial resolution and object shapes [7].

Although the encoder is variable, the backbone was chosen to be ResNet-50, as it had fewer learnable parameters than the Senet and Densenet backbones, which enabled faster training. Also, as this paper was a form of proof-of-concept, the backbone architecture is not too important, as it can always be replaced with a more suitable alternative in the future. The total number of learnable parameters is approx. 68 million [7].

The architecture in [7] took in an RGB image of size 228×304 and produced a depth map at half the scale of 114×152. This format remained largely unmodified for the refinement network, but the first convolution was altered to use a stride of 1 instead of 2 to preserve the dimension of the input.

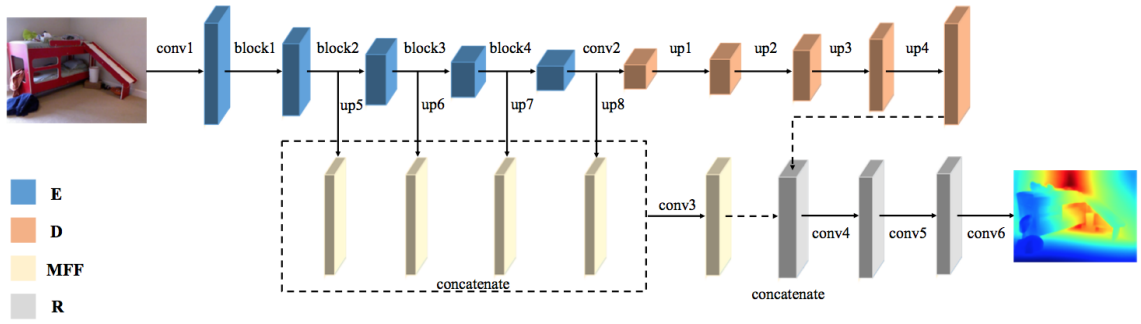

Figure 5: Base Network and Refinement Network Architecture

4.2. Dataset

The NYU-V2 dataset [14] was chosen, as it is one of the only datasets to provide labelled images alongside RGB-D data, allowing for the extraction the ground-truth segmentation masks as inputs. The NYU-V2 dataset provides 1449 labelled images, with 795 training and 654 testing images. With each image being labelled with approximately 30 different objects, this resulted in around 40,000 total image-mask pairs. However, only connected masks, and masks that took up at least 1 percent of the total image space were selected, as these were reasonable consideration for the task of estimating the depth of a photographed object. This resulted in reducing the total number of combinations to approx. 14,000. The images, originally 640×480 in size, were scaled and cropped to 228×304, while the depth maps were scaled and cropped to 114×152, as specified in [7].

Due to the relatively small number of training images, dataset augmentation had to be performed. This augmentation was performed following the guidelines in [7].

- Flip: The RGB and the depth image are both horizon- tally flipped with 0.5 probability.
- Rotation: The RGB and the depth image are both rotated by a random degree $r \in [5,5]$.
- Color Jitter: Brightness, contrast, and saturation values of the RGB image are randomly scaled by $c \in [0.6,1.4]$.

The binary masks followed similar transformations, but due to the bilinear resizing were no longer binary. Thus, an arbitrary but seemingly reasonable threshold of 0.5 was selected to re-binarize the masks. It was observed that the threshold choice did not affect the final outcome, but the

downscaling ensured that the masks did not perfectly capture object boundaries of the ground truth depth.

4.3. User Input

After the base network provided an output, this output was ordinally compared with the ground truth depth map. User clicks were sampled at random proportionally to the number of elements in the mask at levels of $p = 0.05, 0.1, 0.25, 0.5$, where p represents the approx. percentage of points selected for comparison. Figures 6, 7, 8, 9 show what the randomly sampled points might look like in a square mask for the specified p.

To be specific, the value of p only represents an approximate percentage of sampled because the points were determined by masking a uniform sampling. Moreover, points would only be considered if the relative difference between ground truth and predicted output was larger than 25 percent, an arbitrarily threshold that seemed within reason for human perceptive capabilities. This sought to prevent the network from over-compensating where it would not be necessary. Nevertheless, due to the large number of elements in a mask (with the minimum size of 174 pixels) and assumed uniformity of incorrectness, these percentages can by and large be viewed as consistent across images. The benefit of not requiring an exact amount of points was two-fold: firstly, it provided an efficient way of sampling a large number of points in the mask without replacement; secondly, it introduced some robustness to the number of inputted click pairs, hopefully generalizing better to arbitrary numbers of click pair inputs.

Another selection criterion was a single point with the largest discrepancy between the predicted depth and the true depth, as this would be a clear choice for a user when considering refinement. Figure 10 shows an example of such a selection for a masked difference image between predicted and ground truth depth. Then, the element at that given index in the ordinal constraint channel would either be assigned $\{-1, +1\}$ if the predicted depth is relatively closer or farther than the ground truth, respectively. To be specific, if $\max(\frac{di}{gi}, \frac{gi}{di}) > 0.25$, i.e. our user perception threshold has been satisfied between ground truth pixel g_i and predicted depth pixel d_i, then the pixel of the

constraint matrix $P_i = \text{sgn}(\frac{d_i}{g_i} - 1)$. The maximum of relative depths was selected to remove the influence of the denominator selection on the relative difference; otherwise, there would be a bias towards selecting points either closer or farther away from the the ground truth.

This is a fairly easily achievable prediction for a human user, and thus we need not worry too much about the robustness of our predictions as it can be assumed that all the input would be correct (barring mis-clicks or not paying attention). Of course, in general, humans aren't attentive or dexterous enough for pixel-wise predictions, but this research can serve as a baseline for future approaches such as gaussian blobs around a used-defined center.

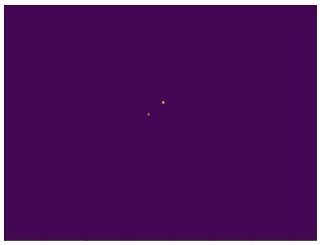

Figure 6: p = 0.01

Figure 7: p = 0.1

Figure 8: p = 0.25

Figure 9: p = 0.5

4.4. Loss Function

The loss function used for training was an extension of the one provided in [7], which considered the sum of depth errors, gradient errors, and normal errors with equal weighting. Equations are provided in the source paper. This loss was then masked so that only differences within the mask would be weighed. It was observed that this loss still produced a relatively smooth image, and did not sufficiently take into account the user clicks. Thus, a new point loss was introduced that considered the depth errors, gradient errors, and normal errors at the user click points. This loss was

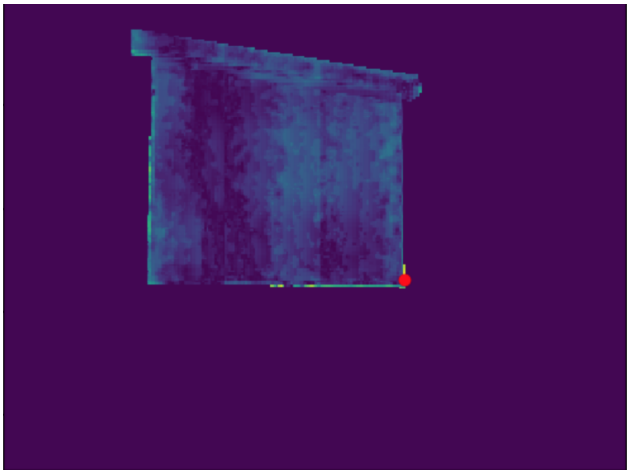

Figure 10: Simulating user click selection on a masked difference map

then summed with the original masked loss with proportionally half the influence. A gaussian loss that produced gaussian blobs around the user click points so that neighbours would be considered (but with less importance) was attempted, but proved too time consuming for full-training and evaluation. Ultimately, the decision was made not to add the point loss to the final loss as it did not improve the refinement.

4.5. Training and Testing

The refinement module was initially trained to learn the identity function on the base output as the sole channel. This ensured a reasonable starting point for the refinement. The weights for the other two channels were then initialized from the standard normal distribution, such that their max and min would roughly align at one order of magnitude lower than the depth input channel to initially prioritize the identity. During training/validation/testing, both the depth dataset and the base output were normalized to a range between zero and one, as non-zero depths made certain calculations easier, and unit-variance standardization did not seem to affect the results. This ensured that relative differences were preserved more accurately by reducing the impact of scaling.

A validation set was chosen to be 20 percent of the training set's size. Training was completed for 1 epoch using the Adam optimizer with a learning rate of 1×10^{-5} and weight decay 1×10^{-4}, as it was observed that training for more epochs did not improve final results. The base network weights were omitted from the back-propagation step, such that only the refinement network weights were

learned to avoid overfitting a dataset already pre-trained on NYU-V2.

Iterative refinement was also considered by training on a RNN of with three iterations. It was observed that this resulted in inferior performance than just training on a single iteration, and so just a single iteration was used for training.

The network was trained using the Pytorch framework. An AWS EC2 instance with a single NVidia Tesla V100-SXM2 GPU was used, taking approximately 20 minutes per epoch to train.

5. Evaluation

Although this project initially started by comparing its performance relative to other interactive models, it was later determined that this was not a feasible task as the model engaged with user input in a rather different way and was more so exploratory than outcome-oriented. In large part, this is due to the fact that the refinement model was not truly iterative. Namely, much like in [12], the refinement model tended to perform best after one iteration, with subsequent iterations decreasing in performance until a plateau was reached.

As a result, the baseline to compare against is the output of the base network of [7], as opposed to better-performing interactive models that also benefit from more complexity. Then, the various techniques explored in this research can serve as a guideline for a future work to exploit and improve the best-performing techniques.

The metrics that was considered are common error metrics in depth estimation, and moreover are already employed in [7]. Denoting the i-th pixel in the predicted depth map as d_i, that same pixel in the ground truth depth map as g_i, and the number of pixels in the depth mask as T, we have the following metrics:

- Root Mean Squared error (RMS): $\sqrt{\frac{1}{T}\sum_{i=1}^{T}(d_i - g_i)^2}$.
- Mean Absolute Error (MAE): $\frac{1}{T}\sum_{i=1}^{T}|(d_i - g_i)|$.
- Mean Relative Error (REL): $\frac{1}{T}\sum_{i=1}^{T}|(d_i - g_i)|/g_i$.
- Thresholded Accuracy: percentage of d_i such that $\max(\frac{d_i}{g_i}, \frac{g_i}{d_i}) = \delta <$ threshold.

The choice to only consider the error within the mask was due to the problem formulation, where

Training Parameters	Results					
	RMS	MAE	REL	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Base Network Output	0.612	0.371	0.133	0.831	0.964	0.991
0	0.650	0.383	0.140	0.819	0.961	0.991
Max point	0.638	0.392	0.136	0.808	0.960	0.990
0.05	0.576	0.373	0.148	0.816	0.979	0.995
0.1	0.536	0.354	0.119	0.870	0.983	0.998
0.25	0.443	0.276	0.100	0.935	0.993	0.998
0.5	0.511	0.349	0.122	0.845	0.994	0.999

Table 1: Refinement Module Results after 1 iteration, with Masked Error

information outside the mask is not important. The distance error metrics correspond to different penalization weighting for the amount of error discrepancy. For instance, the RMS error would penalize larger discrepancies significantly more than MAE error. The first few error metrics are better when tending towards zero, whereas the thresholded accuracy metric is better when tending towards one. Table 1 shows the results on one iteration of the refinement module.

These results clearly indicate the adding user selections can improve the base network. Though one might think at first think that the improved results are due to training with a deeper network. The fact that omitting user constraints and just training with the base output and the segmentation mask/using few ordinal constraints does not improve the base output shows that the ordinal relations here are significantly affecting the output. In fact, using a $p = 25$, the base output is improved by 27.6% with respect to RMS error. Given that there are on average 1055 pixels in a mask, this would amount to 264 user clicks, a seemingly unfeasible amount of clicks for this to be a streamlined process. But we must also take into account that the user perception threshold was set at a relative difference greater than 25%. Approximately 1.5% of masked pixels satisfy this criteria, which means that the user would only have to sample 4 click pairs on average. In contrast, 4 randomly sampled click in [12]'s ADMM network only reduces RMS by 3% from 0.647 to 0.628.

Though the refinement module improves the depth refinement, there are some failure cases. The first failure case results when the predicted depth map is already very good. This results in the ordinal depth constraint matrix being empty due to the high user threshold even with a high p. As a result, the refinement makes a bad prediction, and has difficulty converging on a better solution

unless the mask channel alone provides enough information – this is generally not the case, however. An approach to handle this is to lower the user threshold (say to 10%, though the effects of doing so have not been tested), or to abort the refinement step if the constraint matrix is empty/too sparse as a result of the initial depth refinement being highly accurate, though that could result in the second failure case. Interestingly, the majority of cases on the best performing refinement module had an empty constraint matrix. Though the refinement would generally suffer as a result, the few cases where the P matrix was nonempty seemed to greatly benefit from a refinement, which is likely why the overall testing error went down.

Another failure case is overcompensation. What happens here is that the refinement module swings too far the other way when handling the thresholding. Figures 11, 12, 13 demonstrate this effect. This is potentially the reason why $p = 0.25$ performs slightly better than $p = 0.5$, as dense point clouds will overwhelm an otherwise sensitive refinement module. If this is indeed the case, then the module can be improved in one of four ways (or a combination of these). Firstly, the depth relations can be made ordinal with respect to themselves. What is meant by this is to rank difference according to its magnitude. For example, all discrepancies with user threshold $\leq 10\%$ will be in $\{-1,+1\}$, all discrepancies with user threshold $> 10\%, \leq 20\%$ will be in $\{-2,+2\}$ etc. This is still in the range of human ability (deciding which point pairs need significant as opposed to slight improvement), and will give the network more information to consider as opposed to blindly correct all point pairs by the same amount. Secondly, users should only select pairs with similar magnitudes of error to ensure that corrections are proportionally applied. Thirdly, a clustering algorithm such as K-means can be applied to isolate the most important point pairs and apply these selectively. In fact, if this approach is successful this would suggest that the user should be able to click on many fewer pairs than before. Lastly, a larger user threshold can be chosen to minimize variance within the points, at the cost of running into the first failure case.

Figure 11: Initial depth output (darker is closer)

Figure 12: Ordinal constraints matrix (darker is too close)

Figure 13: Refined depth output (darker is closer)

5.1. multiple iterations

The goal of this project was to be able to successively refine predicted depth maps to minimize some metrics of error. Ultimately, this was not possible. Applying the refinement module for successive iterations would unequivocally result in decreased performance along any metric. The reason for this not fully understood, but the author hypothesizes that the refinement module alters the depth too much at each iteration, and thus it is not possible to converge at local solutions. The observed pattern of large jumps in RMS per iteration confirms this. To solve this, the refinement module would benefit from reducing its weighting after each iteration, so that a local minimum can be achieved, much akin to standard numerical solvers such as SGD.

6. Conclusion

In 2, the following questions were set forth: what is the relationship between click pairs and depth refinement? Is it possible to successively iterate the refinement modules multiple times? Can a CNN be trained to refine the depth map without specifying an optimization problem to solve? How does the addition of a segmentation mask aid in the depth estimation?

We can now attempt to answer these questions formally. This project has shown that there is a non-trivial relationship between ordinal click pairs and depth refinement. By modifying the original concept of ordinal click pairs as intra-image ordinal relations, and considering inter-image ordinal relations between the base output and the ground truth output, this paper demonstrated that a relatively small amount of click pairs can be selected to improve the outputs of an existing state-

of-the-art base network. The mechanism of action is to locally perform the what the constraint tells us, i.e. move the point closer if it should be closer, while simultaneously performing a smoothing operation.

This project did not succeed in implementing multiple iterations of a refinement module, but it did make headway in understanding why this was not viable with the current implementation. Namely, the refinement module did not iteratively reduce the amount by which it adjusted the depth map according to ordinal constraints, which prevented convergence.

As far as comparisons to similar approaches, this refinement stood to perform much worse than state-of-the-art interactive depth estimation, as it failed to leverage the iterative property to achieve convergence. However, not only did it outperform the base network of [7] on average, but it also greatly outperformed the optimization-specified ADMM module in [12] when sampling on average the same amount of click pairs (although the click pairs did represent different information).

The segmentation mask on its own to did not seem to help the depth estimation too much, and the performance actually decreased after training. After all, the loss function was already masked, which meant that the segmentation mask could not provide much contextual information to the scene. Combined with click pairs, however, the network performed fairly well, and was able to correctly capture the shape of the binary mask fairly well when the base output was too smooth. Another limitation was that the binary masks themselves were often a little bit off from the ground truth figure, likely due to the pre-processing steps, which made the underlying assumption of a correct segmentation mask untrue, and this could have produced bias in the results.

In general, this project has shown an interesting proof-of-concept, namely that is possible to learn a refinement module without specifying any optimization problem, and moreover that this is possible with a very simple easy-to-simulate user-accessible input. Nevertheless, in the context of the overall scope of integrating this component with a 2D-3D shape completion mobile app, this project is not readily usable. After all, it is not possible to iteratively refine the depth map like other methods, so it is still very sensitive to the base network output. On top of that, the refinement module does not necessarily improve the base output regardless, which would prove cumbersome

for the aforementioned task.

There is, however, much room for improvement, and simple improvement at that. A lot of decisions had to be made somewhat arbitrarily, or without full consideration of scope, due to limited computing power and time. As a result, many hyper parameters, such as user thresholding, [7]'s loss function hyperparameters, p value etc. have not been fully explored and optimized. This can be left for future work. Other work would consider testing out this network on other databases, though currently this is subject to the availability of ground-truth segmentation masks. This paper has already mentioned techniques such as ranking click pairs by error magnitude to provide more information to the network, and implementing the module as an RNN, which the author believes will be the most relevant endeavours for future work. It is also possible that including the RGB image for training would help the refinement preserve its understanding of the scene as the refined outputs change.

References

[1] [Online]. Available: https://www.apple.com/business/site/docs/FaceID_Security_Guide.pdf

[2] and, , S. Cohen, B. Price, and A. Yuille, "Towards unified depth and semantic prediction from a single image," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 2800–2809.

[3] J. Chen, H. Yuan, and J. Song, "Error-tolerant interactive 2d-to-3d conversion using local consistency," in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, July 2018, pp. 30–33.

[4] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," *CoRR*, vol. abs/1604.03901, 2016. [Online]. Available: http://arxiv.org/abs/1604.03901

[5] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *CoRR*, vol. abs/1411.4734, 2014. [Online]. Available: http://arxiv.org/abs/1411.4734

[6] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep ordinal regression network for monocular depth estimation," *CoRR*, vol. abs/1806.02446, 2018. [Online]. Available: http://arxiv.org/abs/1806.02446

[7] J. Hu, M. Ozay, Y. Zhang, and T. Okatani, "Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries," *CoRR*, vol. abs/1803.08673, 2018. [Online]. Available: http://arxiv.org/abs/1803.08673

[8] S. Iizuka, Y. Endo, Y. Kanamori, J. Mitani, and Y. Fukui, "Efficient depth propagation for constructing a layered depth image from a single image," *Computer Graphics Forum*, vol. 33, no. 7, pp. 279–288, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12496

[9] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," *CoRR*, vol. abs/1606.00373, 2016. [Online]. Available: http://arxiv.org/abs/1606.00373

[10] A. Lopez, E. Garces, and D. Gutierrez, "Depth from a Single Image Through User Interaction," in *Spanish Computer Graphics Conference (CEIG)*, A. Munoz and P.-P. Vazquez, Eds. The Eurographics Association, 2014.

[11] A. Mousavian, H. Pirsiavash, and J. Kosecka, "Joint semantic segmentation and depth estimation with deep convolutional networks," *CoRR*, vol. abs/1604.07480, 2016. [Online]. Available: http://arxiv.org/abs/1604.07480

[12] D. Ron, K. Duan, C. Ma, N. Xu, S. Wang, S. Hanumante, and D. Sagar, "Monocular depth estimation via deep structured models with ordinal constraints," 09 2018, pp. 570–577.

[13] R. Rzeszutek, R. Phan, and D. Androutsos, "Semi-automatic synthetic depth map generation for video using random walks," in *2011 IEEE International Conference on Multimedia and Expo*, July 2011, pp. 1–6.

[14] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760.

[15] J. T. Todd and J. F. Norman, "The visual perception of 3-d shape from multiple cues: Are observers capable of perceiving metric structure?" *Perception & Psychophysics*, vol. 65, no. 1, pp. 31–47, Jan 2003. [Online]. Available: https://doi.org/10.3758/BF03194781

[16] H. Yuan, S. Wu, P. Cheng, P. An, and S. Bao, "Nonlocal random walks algorithm for semi-automatic 2d-to-3d image conversion," *IEEE Signal Processing Letters*, vol. 22, no. 3, pp. 371–374, March 2015.

[17] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," 2013.

[18] D. Zoran, P. Isola, D. Krishnan, and W. T. Freeman, "Learning ordinal relationships for mid-level vision," 12 2015, pp. 388–396.