Monocular Depth Estimation using Deep Learning

183.663 Deep Learning for Visual Computing - 2021S
Group 22

Benjamin Schwendinger (e01225371) and Christian Stippel (e11778254)
https://github.com/ben-schwen/dlvc_group17

June 30, 2021

Introduction

Estimating depth information from images is an important task in computer vision which subsequently can be used in processes of Simultaneously Localization and Mapping (SLAM) or Visual Odometry (VO). While geometric-based approaches are historically dominating these fields [2, 7], recent and rapid advances in using Deep Learning for depth estimation have been made [3, 6, 5]. Inferring depth information from a single image (monocular depth estimation) has the advantages of lower hardware costs and less calibration efforts as opposed to stereo and visual-inertial odometry methods which achieve higher performances due to scale drift and low robustness. Harnessing the incredible power of deep learning the task at hand is to learn a mapping from a given RGB image to a tensor of depth values in a real end-to-end manner.

Data set and metrics

Since the benchmarked models are trained on various relative depth data sets like ReDWeb [11], NYU Depth Dataset V2 [10], etc. we created our own data set for benchmarking purposes to prevent any information leakage. To record not only the RGB values but also the true depth map of an image we used an Intel RealSense Depth Camera D435 which is a stereo camera with prior known focal length and baseline. The idea behind this projection for stereo images is visualized in Figure 1 below.

Our final data set consists out of 11921 RGB and depth images in 640 x 320 format. Examples of this data set together with their corresponding ground truth depth images are shown in Figure 2.

Using a stereo camera there are points which are occluded from one view of the camera, thus their depth values become undefined, since no disparity can be measured. These values are assigned with 0 from the Intel RealSense framework and need extra caution.

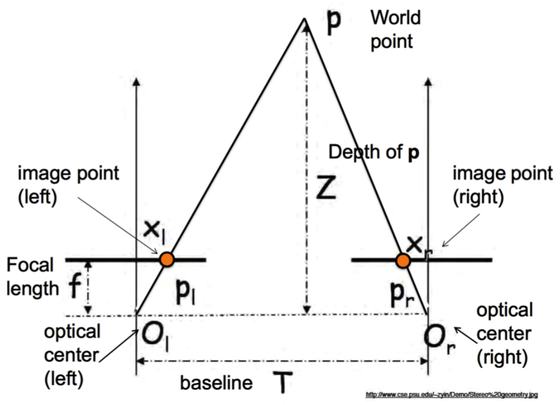


Figure 1: Example of two parallel calibrated cameras

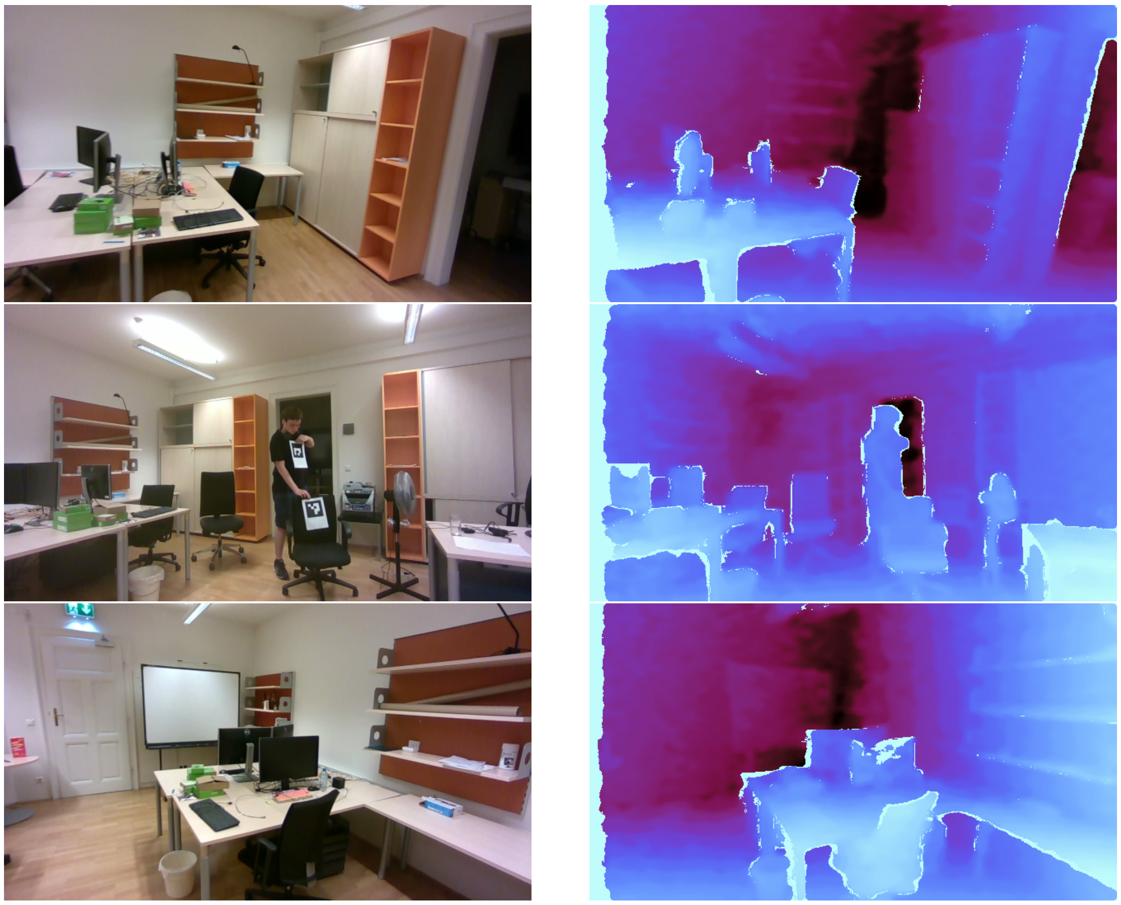


Figure 2: Example depth images from Intel RealSense Depth Camera D455

ArUco Markers

For qualitative testing and camera calibration we used ArUco markers [4] which are black and white visualizations of binary matrices. The inner matrix encodes the marker id together with a wide black border. See also Figure 3 for an example of such a marker. These fiducial markers can be used for a camera pose estimation. Camera calibration is done by taking pictures of an ArUco board from multiple view points in order to adjust for radial and tangential lens distortion, as well as finding the intrinsic camera parameters to calculate the geometry of the scenery. Skipping the calibration part or using bad (too few pictures) calibration values leads to a distortion of the true scale of the scenery.

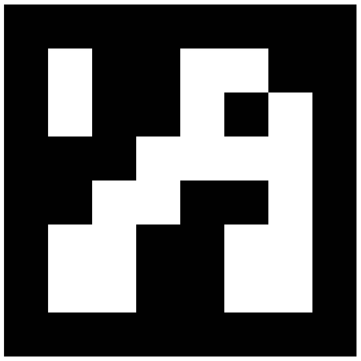


Figure 3: ArUco marker with id 23

Anchor points

Since most monocular depth estimation networks only return relative depth estimations instead of true depth estimations "known" anchor points can be used for scaling the results of these networks. The anchor point itself is the middle point of an ArUco marker which can, as already mentioned, be used for camera pose estimation. For a justification of this approach we used images with two ArUco markers where marker 0 represents the known anchor point and marker 1 displays the predicted depth value of the AdaBins network together with the true value as determined with the marker itself. We highly recommend to watch our recorded videos at `https://youtu.be/UrvkL1H2eDY` which tries to visualize this idea even better.

Executive summary

Essentially, our overall approach can by summarized by the following steps:

1. Create data sets with a ground truth
2. Estimate the depth:
    (a) with AdaBins to obtain true scale depth values
    (b) with MiDas to obtain the inverse depth and try to work with it
3. Compare them quantitatively in regards of RMSE log
4. Evaluate and explain AdaBins qualitatively with ArUco patterns

Used models

The relevant publications as well as the source-code repositories of the used models are displayed in Table 1.

Model name	Repository	Paper
AdaBins	`https://github.com/shariqfarooq123/AdaBins`	[1]
MiDaS	`https://github.com/intel-isl/MiDaS`	[9, 8]

Table 1: Benchmarked models

AdaBins

AdaBins is a deep learning model for monocular depth estimation. It consists of a convolutional encoder-decoder as well as a AdaBins module. The AdaBins module itself is a transformer inspired architecture which estimates the bin-widths of the depth distribution of a given image as well as Range-Attention-Maps which can later be used for pixel-level depth computation.

MiDaS

MiDaS (also sometimes called DPT by the authors) is a vision transformer which uses transformers as a encoder in its encoder-decoder architecture instead of the classically used convolutional networks. On a high level view a given image is split into multiple patches which are treated as a bag-of-words representation of the given image. Here a single patch is considered a token. The transformer itself transform this set of tokens using sequential blocks of multi-headed self-attention. Interestingly, this network achieves state-of-the-art results compared to convolutional encoder-decoder approaches of similar network size.

Computational results

Evaluation metrics

In order to evaluate the performance of the various models we calculated the RMSE log. This metric is defined by

- RMSE $\log(y, \hat{y}) = \sqrt{\frac{1}{|N|} \sum_{i \in N} || \log(y_i) - \log(\hat{y}_i)||^2}$

where y_i is the measured ground truth depth value of pixel i and $\hat{y}_i$ the corresponding predicted depth. Moreover, N denotes the set of indices of all pixels of a given image and $|N|$ the cardinality of this set.

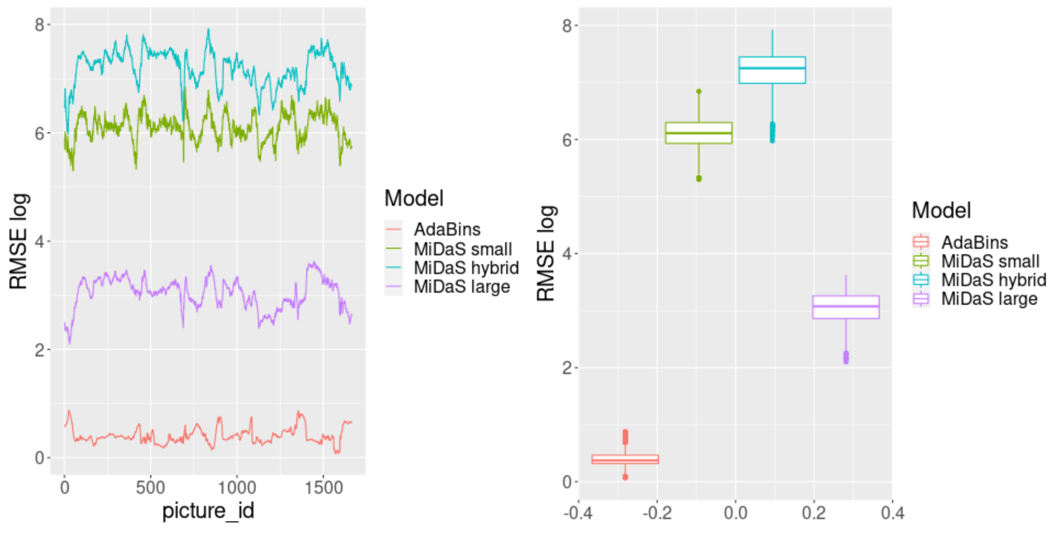
Hardware

All tests were carried out using:

1. AMD Ryzen Threadripper 1920X 12-Core Processor
2. GeForce RTX 2080 Ti Rev.

Results

We benchmarked the previously described models by evaluating their performances on a part of our data set which we call ”Office” and which consists out of 1664 RGB images in a 640 × 360 format. Here we experienced an outstanding performance of AdaBins compared to different sized versions of MiDaS. The performance regarding a single image can be seen in part (a) of Figure 4 below, while the overall performance across all images is in depicted in part (b). Interestingly, we can see that the performance of MiDaS seems to be more stable, across the time lapse of the different images, while a higher relative standard deviation is occurring at AdaBins.



(a) RMSE comparison per image and model (b) Overall model comparison

Figure 4: Performance of the models on our Office data set

Model	RMSE (mean)	RMSE (sd)
AdaBins	**0.4049**	**0.1442**
MiDaS small	6.1046	0.2537
MiDaS hybrid	7.1944	0.3206
MiDaS large	3.0433	0.2863

Table 2: Bencharking results of examined models on Office data set

An example of side by side comparison of the different models is displayed in Figure 5. The full side by side comparison can be found at `https://youtu.be/x7cvzDrKnns` which we also greatly recommend to watch.

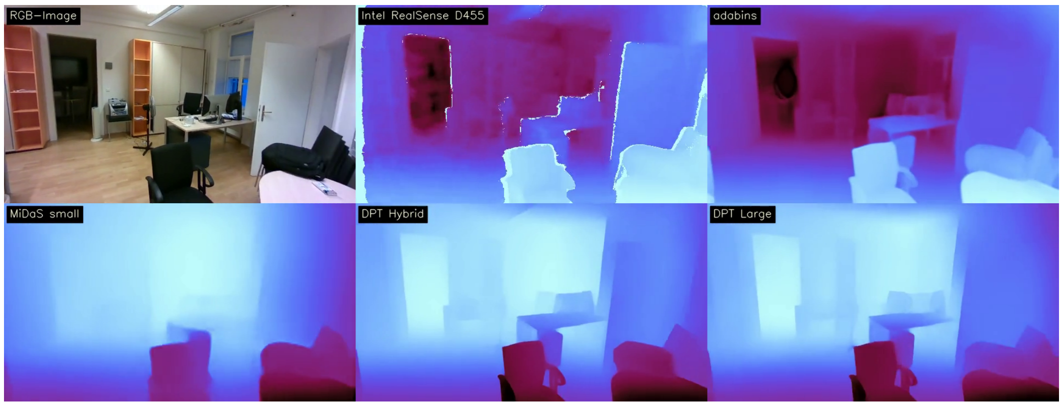


Figure 5: Example of side by side comparison depth maps

Encountered problems

Calibration

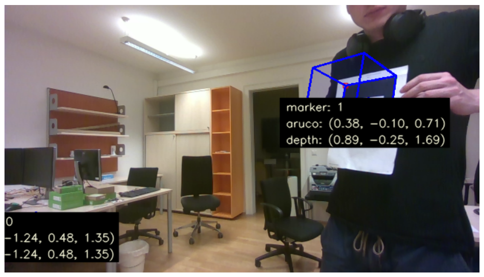
Using too few pictures or using not heterogenous enough pictures for camera calibration leads to a distortion of the true scale of the captured scenery and as a result also to wrong values of the camera position estimation. This can simply be countered by using more and more heterogenous (regarding distances and angles) pictures for the calculation of the distortion as well as the intrinsic camera parameters.

Relative depth

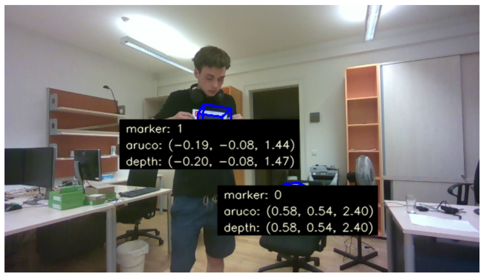
Since many depth estimation networks only produce relative depth maps by squeezing their first estimates to a 0..1 range with non-linear transformations we were not able to obtain the true depth directly from the networks. This issue could possible be overcome by introducing additional anchor points but this lies out of the scope of this work. In the presented work, we tried scaling and shifting the depth by constant factors as well as inverting the depth maps (as suggested in original repository of MiDaS), however, as seen in Figure 4 the networks with no true scaling still behaved much worse when compared to AdaBins.

Bad anchors

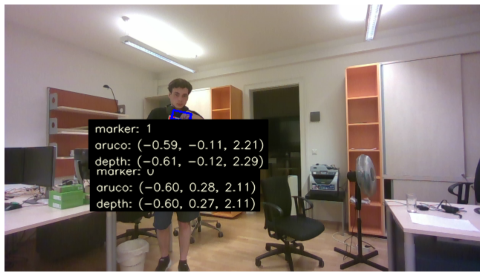
Since the anchor points determine the scaling of our relative depth estimation to convert it into a true depth estimation, we empirically experienced the importance of choosing an appropriate place for our used anchor points. As depicted in Figure 6 below we can see that the anchor point can have a strong influence on the quality of our scaled prediction. On the right hand side we can see how a bad placed anchor distorts the results of the depth estimation in comparison to a well placed on the left hand. Moreover, we experienced chairs as good anchor places which we explain by the multiple reoccurring appearances of chairs in the data sets that have been used for originally training the used models. This dilemma with bad anchor points is also visualized in `https://youtu.be/09Sevrv2lBM` where the anchor point is at first at a bad anchoring place and after $\sim$ 20 seconds switched to a better anchoring place.



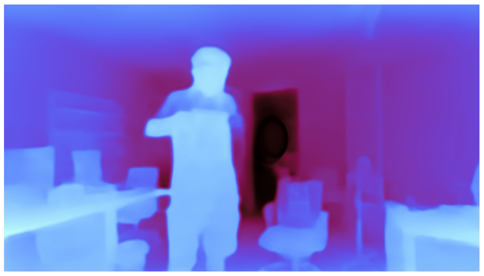
(a) An example of a bad placed anchor


marker: 1
aruco: (−0.19, −0.08, 1.44)
depth: (−0.20, −0.08, 1.47)
marker: 0
aruco: (0.58, 0.54, 2.40)
depth: (0.58, 0.54, 2.40)

(b) An example of a well placed anchor


marker: 1
aruco: (−0.59, −0.11, 2.21)
depth: (−0.61, −0.12, 2.29)
marker: 0
aruco: (−0.60, 0.28, 2.11)
depth: (−0.60, 0.27, 2.11)

(c) ArUco marker on the same surface



(d) Depth map of well placed anchor

Figure 6: Examples representing the strong influence of choosing appropriate anchor points

Conclusion

Monocular depth estimation is a very interesting research area with an active research community and already quite promising state of the art results. While the performance of relative depth estimating networks is quite astonishing, there is still a lot to be done for obtaining and estimating true depth values also with regards to edge and mobile computing and maybe even without the use of anchoring points. Overall we can say that the targeted project scope was probably too ambitious. While the reached results coincide with our initial vision, the workload behind all of it was poorly underestimated. To make our work as reproducible as possible we also included a small README at `https://github.com/ben-schwen/dlvc_group17`.

Further work

Future work could include the estimation of real depth by using the inverse depth map in combination with a previously given depth map using techniques similar to already established practices used in inpainting. Since the data set was taken while moving the camera position, additional structure from motion depth estimation techniques could possible by leveraged as well. The occurring gaps of the Intel RealSense depth image (arising as white points in the coloring of the depth map) could also be filled by interpolating the neighborhood values of these critical points or interpolating previous obtained results of the depth map.

Acknowledgements

We are gradually thankful to CognitiveXR for lending us a stereo camera for the creation of our data set. Without their kindness and trust this work would not have been possible.

References

[1] 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*, pages 4009–4018, 2021.

[2] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[3] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2002–2011, 2018.

[4] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[5] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 270–279, 2017.

[6] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6647–6655, 2017.

[7] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[8] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *arXiv preprint arXiv:2103.13413*, 2021.

[9] Rene Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[10] 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.

[11] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. Monocular relative depth perception with web stereo data supervision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.