Learning structure-from-motion from motion

Clément Pinard[1,2], Laure Chevalley[2], Antoine Manzanera[1], and David Filliat[1]

[1] ENSTA ParisTech
Computer Science and System Engineering Department
Palaiseau, France
{clement.pinard, antoine.manzanera, david.filliat}@ensta-parsitech.fr
[2] Parrot, Paris, France
laure.chevalley@parrot.com

Abstract. This work is based on a questioning of the quality metrics used by deep neural networks performing depth prediction from a single image, and then of the usability of recently published works on unsupervised learning of depth from videos. These works are all predicting depth from a single image, thus it is only known up to an undetermined scale factor, which is not sufficient for practical use cases that need an absolute depth map, i.e. the determination of the scaling factor. To overcome these limitations, we propose to learn in the same unsupervised manner a depth map inference system from monocular videos that takes a pair of images as input. This algorithm actually learns structure-from-motion *from motion*, and not only structure from context appearance. The scale factor issue is explicitly treated, and the absolute depth map can be estimated from camera displacement magnitude, which can be easily measured from cheap external sensors. Our solution is also much more robust with respect to domain variation and adaptation via fine tuning, because it does not rely entirely on depth from context. Two use cases are considered, unstabilized moving camera videos, and stabilized ones. This choice is motivated by the UAV (for Unmanned Aerial Vehicle) use case that generally provides reliable orientation measurement. We provide a set of experiments showing that, used in real conditions where only speed can be known, our network outperforms competitors for most depth quality measures. Results are given on the well known KITTI dataset [1], which provides robust stabilization for our second use case, but also contains moving scenes which are very typical of the in-car road context. We then present results on a synthetic dataset that we believe to be more representative of typical UAV scenes. Lastly, we present two domain adaptation use cases showing superior robustness of our method compared to single view depth algorithms, which indicates that it is better suited for highly variable visual contexts.

1 Introduction

Scene understanding from vision, in particular depth estimation, is a core problem for autonomous vehicles.

One could train a system for depth from vision with supervised learning on an offline dataset which features explicit depth measurement, such as KITTI [1], but even setting up such recording devices can be costly and time demanding, which can limit the amount of data the system can be trained on.

As a consequence, in this paper we are specifically interested in **unsupervised** learning of depth from images using machine learning optimization techniques.

Training to infer the depth of a scene and one's ego-motion is a problem for which recent work has been successfully done with no supervision, leveraging uncalibrated data solely from RGB cameras, but to our knowledge, all of them infer depth from a single image [2–6]. On the contrary, our methods tries to deduce it from multiple frames, using motion instead of context.

UAV navigation, which is one of our favorite use cases, is very specific compared to other ego motion videos. Its two main characteristics are the availability of orientation and the high variability of the visual context:

- An UAV relies on inertial data to maintain its position and the current market for UAVs allows to get high quality video stabilization even for consumer products. As such, orientation of any frame can be assumed to be well estimated.
- Compared to videos acquired from any other vehicle, UAV scenes are very heterogeneous. Unlike a camera fixed to a car, altitude can vary widely and quickly, along with velocity, orientation, and scene visual layout, which context can be hard to figure out with only one frame.

Hence, we propose an unsupervised scene geometry learning algorithm that aims at inferring a depth map from a sequence of images. Our algorithm works with stabilized *and* unstabilized videos, the latter requiring in addition to the depth estimator network an orientation estimator that can bring back digital stabilization.

Our algorithm outputs a depth map assuming a constant displacement, this solves the scale factor incertitude simply by knowing ego-motion speed. This allows a straightforward real conditions depth inference process for any camera with a known speed such as the one of the supporting cars or UAVs.

2 Related Work

First works trying to compute a depth map from images using machine learning can be found as early as 2009 [7]. Whether from multiple frames or a single frame, these techniques have shown great generalization capabilities, especially using end-to-end learning methods such as convolutional neural networks.

2.1 Supervised Depth Networks

Most studied problems for supervised depth learning use a stereo rig along with corresponding disparity [8, 9], thanks to dedicated datasets [1, 10]. For unconstrained monocular sequences, DepthNet and DeMoN [11, 12] are probably the

works closest to ours. Using depth supervision, these networks aim to compute a depth map with a pair of images from a monocular video.

The first network explicitly assumes a fixed displacement magnitude and a stabilized video, and only outputs depth, while the second one also outputs a pose, from which translational component is trained to be of constant magnitude. Both methods easily solve the scale factor problem when the camera speed is known. Our main goal here is to achieve the same operation, but with unsupervised training.

2.2 Unsupervised Depth Learning

Most recent works on unsupervised training networks for computing depth maps use differentiable bilinear warping techniques, first introduced in [13]. The main idea is trying to match two frames using a depth map and a displacement. The new loss function to be minimized is the photometric error between the reference frame and the projected one. Depth is then indirectly optimized. Although sensitive to errors coming from occlusions, non Lambertian surfaces and moving objects, this optimization shows great potential, especially when considering how little calibrated data is needed.

For instance [4,14,15] use stereo views and try to reconstruct one frame from the other. This particular use case for depth training allows to always consider the same displacement and rigid scenes since both images are captured at the same time and their relative poses are always the same. However, it constraints the training set to stereo rigs, which are not as easy to set up as a monocular camera.

When trying to estimate both depth and movement, [2,3,6,16] also achieved decent results on completely unconstrained ego-motion video. One can note that some methods [2] are assuming rigid scenes although the training set does not always conform to this assumption. The other ones try to do without this assumption by computing a residual optical flow to resolve the uncertainty from moving objects.

[5] explicitly considered non rigid scenes by trying to estimate multiple objects movements in the scene, to begin with the motion of the camera itself, which allowed them to deduce a flow map, along with the depth map. The reader is referred to the work of Zhou *et al* [2] for a more complete vision of the field, as all other works are actually built on this fundamental basis.

3 Single Frame Prediction *vs* Reality

As already mentioned, in the current state of the art of learning from monocular footage, depth is always inferred by the network using a single image. Indeed, it is mentioned in [2] that feeding multiple frames to a network did not yield better results. This may be due to the fact that for particular scenarii with very typical geometric and photometric contexts such as in-car view of the road, depth seems easier to get from the visual layout than from the motion.

Because of the single frame context, current depth quality measurements, originally introduced by Eigen *et al* [17], expects a relative depth map up to a scale factor. This is problematic, as they completely ignore the scale factor uncertainty, and thus rely on estimating the scale factor as the ratio of the medians of network's output and groundtruth. This is then representative to an ideal use case where the median of an unknown depth map has to be available, which is clearly unrealistic.

One can try to overcome these limitations by figuring out the scale factor with several solutions:

- Measuring depth, at least in one point, with additional sensor such as LiDAR, Time-of-Flight or stereo cameras. This is not a trivial solution and it needs integration, as well as precise calibration.
- Assuming depth consistency across training and testing dataset. This can be particularly useful in datasets like KITTI [1], where the camera is always at the same height and looking at the floor from the same angle, but it is irrelevant on a dataset with high pose variability, *e.g.* UAV videos, and such assumptions will fail.

Thankfully, those techniques do not only predict depth, but also ego-motion. An other network tries to compute poses of frames in a sequence. Considering that the uncertainties about depth in one hand and pose in the other hand are consistent, the scale factors for depth and pose are theoretically the same. As a consequence, depth scale factor can be determined from the ratio between translation estimation, and actual translation measurement, which is already available on cars and UAVs.

A new quality measurement can then be designed, by slightly modifying the already prevalent ones. This new measurement is not relative anymore, it computes actual depth errors, and is more representative of a real application case. This will be denoted in tables 1, 2 and 3 by the scale factor column. When using standard relative depth measurement, the indication *GT* (for *Ground Truth*) is used, when using translation magnitude, the letter *P* (for *Pose*) is used.

4 Approach

Inspired from both [2] and [11], we propose a framework for training a network similar to DepthNet, but from unlabeled video sequences. The approach can be decomposed into three tasks, as illustrated by Fig. 1:

- From a certain sequence of frames $(I_i)_{0 \leq i < N}$, randomly choose one target frame I_t and one reference frame I_r, forming a pair to feed to DepthNet.
- For each $i \in [\![0, N[\![$, estimate pose $\widehat{T}_{t \to i} = (\widehat{R}_i, t_i)$ of each frame I_i relative to the target frame I_t, and compensate rotation of reference frame I_r to I_r^{STAB} before feeding it to DepthNet, leading to the same situation as original DepthNet, fed with stabilized inputs in [11]. As discussed in Section 1, when considering UAV footage, rotation can be supervised.

- Compute the depth map which for presentation purpose will be denoted ζ. $\text{DepthNet}(I_r^{\text{STAB}}, I_t) = \zeta(I_r^{\text{STAB}}, I_t)$
- Normalize the translation to constrain it so that the displacement magnitude t_r with respect to I_r, is always the same throughout the training. This point is very important, in order to guarantee the equivariance between depth and motion, imposed by the original DepthNet training procedure.
- As the problem is now made equivalent to the one used in [2], perform a photometric reprojection of I_t to every other frame I_i, thanks to depth ζ_t of I_t and poses $\widehat{T}_{t\to i}$ computed before, and compute loss from photometric dissimilarity with I_i.

The whole reprojection process can be summarized by Eq. 1. where K denotes the camera intrinsic, p_t homogeneous coordinates of a pixel in frame I_t and $\zeta_t(p_t)$ is the depth value of the pixel outputted by DepthNet. p_t^i are homogeneous coordinates of that very pixel in frame I_i. To get the equivalent pixel in coordinate p_t^i, I_i pixels are bilinearily interpolated.

Our algorithm, although relying on very little calibration, needs to get consistent focal length. This is due to the frame difference being dependent to focal length. However, this problem is easily avoided when training on sequences coming from the same camera. Also, as shown by Eq. 1, camera intrinsic matrix K needs to be known to compute warping and subsequent photometric reprojection loss properly. In practice, assuming optical center in the center of the focal plane worked for all our tests; this is corroborated by tests done by [6] where they used uncalibrated camera, only knowing approximate focal length.

$$\begin{aligned} \forall i \in [\![0, N[\![\, , p_t^i &= K\widehat{T}_{t\to i}\left(\zeta_t(p_t)K^{-1}p_t\right) \\ \widehat{T}_{t\to i} &\begin{cases} \mathbb{R}^3 \to \mathbb{R}^3 \\ X \;\mapsto \widehat{R}_i X + t_i \end{cases} \end{aligned} \tag{1}$$

4.1 Pose Estimation

PoseNet, as initially introduced by Zhou *et al* [2] is a classic fully convolutional neural Network that outputs 6 DoF transformation pose descriptor for each frame. Output poses are initially relative to the last frame, and then compensated to be relative to the pose of the target frame. This way, PoseNet output is not dependent on the index of the target frame. Besides, computing by default with respect to the last frame makes the inference much more straightforward, as in real condition, target frame on which depth is computed should be the last of the sequence, to reduce latency.

4.2 Frame stabilization

In order to cancel rotation between target and reference frame, we can apply a warping using rotation estimation from PoseNet. When considering a transformation with no translation, Eq. 1 no longer depends on the depth of each

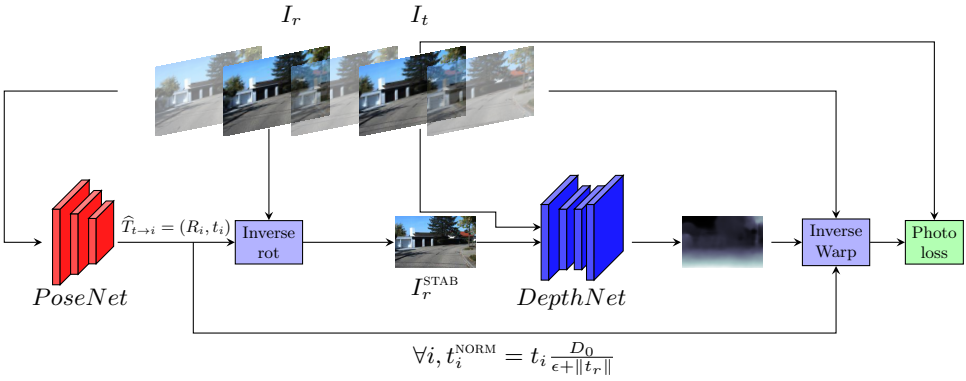

Fig. 1. General workflow architecture. *target* and *ref* indices (t and r) are chosen randomly for each sequence (even within the same batch); output transformations of PoseNet are compensated so that $\widehat{T}_{t\to t}$ is identity. D_0 is a fixed nominal displacement.

pixel, and becomes Eq. 2. As such, we can warp the frame to stabilize it using orientation estimation from PoseNet, before computing any depth.

$$p_t^r = KR_rK^{-1}p_t \tag{2}$$

As mentioned in Section 1, UAV footages are either stabilized or with a reliable estimated orientation from inertial sensors. This information can be easily leveraged in our training workflow to supervise pose rotation, giving in the end only translation to estimate to PoseNet. In addition, when running in inference, no pose estimation is needed, and only DepthNet is used. A similar algorithm as Pinard *et al* [18] can then be used to estimate absolute depth maps at a relatively low computational cost.

4.3 Depth computing and pose normalization

Thanks to the close relation between distance and optical flow for stabilized frames, [11] proposed a network to compute depth from stabilized videos. As depth is then provided assuming a constant displacement magnitude, the pose of the reference frame must be normalized to correspond to that magnitude. As such, to get consistent poses throughout the whole sequence, we apply the same normalization ratio, as shown in Fig. 1:

$$\forall i, t_i^{\text{NORM}} = t_i \frac{D_0}{\epsilon + \|t_r\|} \tag{3}$$

The main drawback of normalizing translations is the lack of guarantee about absolute output values. Since we only consider translations relatively to the reference, translations are estimated up to a scale factor that could be - when

they are very large - leading to potential errors for rotation estimation, or - when they are very close to 0 - leading to float overflow problems. To overcome these possible issues, along with classic L_2 regularization to avoid high values, we add a constant value ϵ to the denominator. The normalization is then valid only when $\epsilon \ll \|t_r\|$.

4.4 Loss functions

Let us denote $\widehat{I_i}$ as the inverse warped image from I_i to target image plane by p_t^i and $\|\cdot\|_1$ the L_1 norm operator (corresponding here to the mean absolute value over the array). For readability, we contract $\zeta(I_r, I_t)$ into simply ζ.

The optimization will then try to minimize the dissimilarities between the synthesized view $\widehat{I_i}$ and original frame I_t. As suggested by [4], raw pixel difference can be coupled with structural similarity (SSIM) [19] maximization, in order to be robust to luminosity changes, either from camera auto-exposition or from non Lambertian surfaces. SSIM computation is detailed Eq. 4, where μ and σ are the local mean and variance operators, estimated by convolving the image with Gaussian kernels of size 3×3. $C_1 = 0.01$ and $C_2 = 0.09$ are two constants. Note that the use of convolution in SSIM increases the receptive field of the loss function with respect to the L_1 distance.

$$\mathrm{SSIM}(I_t, I_i) = \frac{(2\mu_{I_t}\mu_{I_i} + C_1) + (2\sigma_{I_t I_i} + C_2)}{(\mu_{I_t}^2 + \mu_{I_i}^2 + C_1)(\sigma_{I_t}^2 + \sigma_{I_i}^2 + C_2)} \tag{4}$$

Our photometric loss $\mathcal{L}_p$ is then a mixture of the two, α being an empirical weight.

$$\mathcal{L}_p = \sum_i \|\widehat{I_i} - I_t\|_1 - \alpha \mathrm{SSIM}(\widehat{I_i}, I_t) \tag{5}$$

Along with frames dissimilarity, in order to avoid divergent depth values in occluded or low textured areas, we add a geometric smooth loss that tries to minimize depth relative Laplacian, weighted by image gradients. Also, contrary to single frame network, depthnet output ζ here is not normalized. Thus, we must scale it according to its mean value. The fraction here represents pixel wise division, ∇ and Δ are the gradient and Laplacian operators respectively, obtained by 3×3 convolutions.

$$\mathcal{L}_g = \left\| \frac{|\Delta\zeta|}{\|\nabla I_t\|} \right\|_1 \times \frac{1}{\|\zeta\|_1} \tag{6}$$

Finally, we apply this loss to multiple scales s of DepthNet outputs, multiplied by a factor giving more importance to high resolution, and our final loss becomes

$$\mathcal{L} = \sum_s \frac{1}{2^s} \left(\mathcal{L}_p^s + \lambda \mathcal{L}_g^s\right) \tag{7}$$

where λ denotes an empirical weight.

5 Experiments

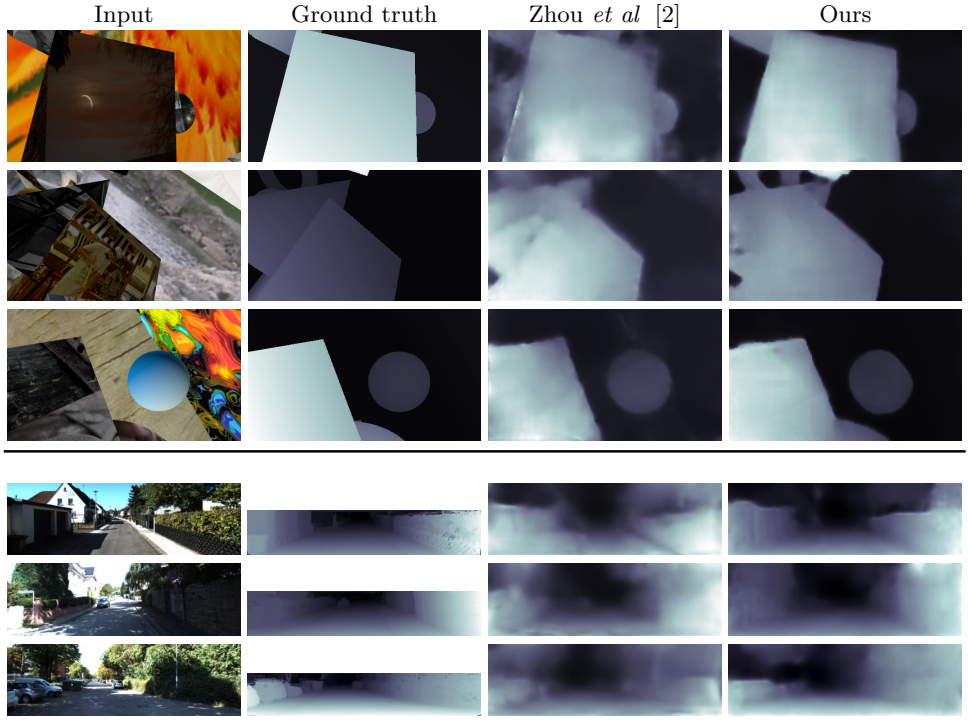

Fig. 2. Comparison between our method and Zhou *et al* [2] on updated Still Box [11] and KITTI [1].

5.1 Training datasets

Our experiments were made on three different datasets. KITTI is one of the most well known datasets for training and evaluating algorithms on multiple computer vision tasks such as odometry, optical flow or disparity. It features stereo vision, LiDAR depth measures and GPS / RTK coupled with IMU for camera poses. During training we only used monocular frames and IMU values when supervising with orientation. We used LiDAR for evaluation. We applied the same training/validation/test split as [2]: about 40k frames for training and 4k for evaluation. We also discarded of the whole set scenes containing the 697 test frames from the Eigen [17] split. We also constructed a filtered test set with the same frames as Eigen, but discarding 69 frames whose GPS position uncertainty was above 1 m. This set was used when displacement data was needed.

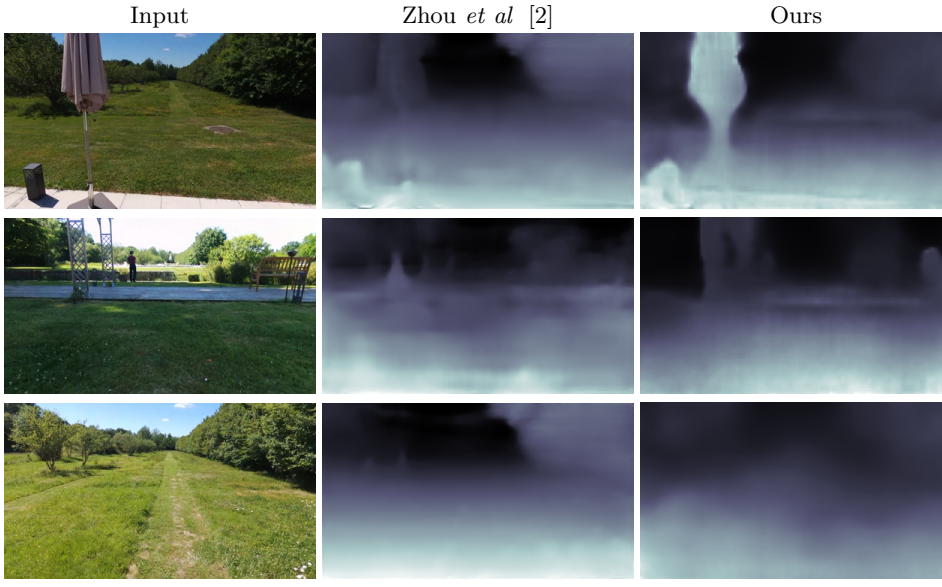

Fig. 3. Subjective comparison of disparity maps between Zhou [2] and our method on a small UAV dataset.

Fig. 4. Some failure cases of our method on KITTI. First column is a detail of a larger image. The foreground car is moving forward and it's detected as far away, while the background car is moving toward us and is detected as close. Second column is a poorly textured road.

We also conducted experiments on an updated version of Still Box, used in [11], in which we added random rotations (*i.e.* we draw an initial rotation speed that remains the same through the sequence). This dataset features synthetic rigid scenes, composed of basic 3d primitives (cubes, spheres, cones and tores) randomly textured using images scrapped from *Flickr*. In this dataset, depth is difficult to infer from context, as shapes have random sizes and positions. Camera's movement is constrained to constant velocity (translation and rotation) throughout scenes of 20 pictures. The dataset contains 1500 training scenes and 100 test scenes, *i.e.* 30k training frames and 2k validation frames.

Finally, we trained our network on a very small dataset of UAV videos, taken from the same camera the same day. We used a Bebop2 drone, with 30fps videos, and flew over a small area of about one hectare for 15 minutes. The training set contains around 14k frames while the test set is a sequence of 400 frames. This dataset is not annotated, and only subjective evaluation can be done.

5.2 Implementation details

DepthNet is almost the exact same as the one used in Pinard *et al* [11]. Its structure mainly consists of two components: the encoder and the decoder parts. The encoder follows the basic structure of VGG [20]. The decoder is made up of deconvolution layers to bring back the spatial feature maps up to a fourth of the input resolution. To preserve both high semantics and rich spatial information, we use skip connections between encoder and decoder parts at different corresponding resolutions. This is a multi-scale technique that was initially done in [21]. The main difference between DepthNet and our network is the ELU function [22] applied to last depth output instead of identity.

PoseNet is the same as [2] which contains 8 convolutional with a stride of 2 layers followed by a global average pooling layer before final prediction. Every layer except the last one are post processed with ReLU activation [23].

We used PyTorch [24] for all tests and trainings, with empirical weights $\lambda = 3$ and $\alpha = 0.075$. We used Adam optimizer [25] with learning rate of 2×10^{-4} and $\beta_1 = 0.9$, $\beta_2 = 0.999$.

5.3 Quality measurements and comparison with other algorithms

In addition to using standard measurements from [17], our goal is to measure how well a network would perform in real conditions. As stated in Section 3, depth map scale factor must be determined from reasonable external data and not from explicit depth ground truth.

We thus compare our solution to [2] where the output is multiplied by the ratio between estimated displacement from PoseNet and actual values. For KITTI, displacement is determined by GPS RTK, but as we only need magnitude, speed from wheels would have been sufficient. When training was done with orientation supervision, we stabilized the frames before feeding them to DepthNet.

To test our method on our small UAV dataset, we first did a training on updated Still Box, then an unsupervised fine tuning. Likewise, when using Zhou *et al* method [2], we pretrained on KITTI before fine tuning on our video.

5.4 Quantative training results

Method	training set	scale factor	supervision	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen *et al* [17] Coarse	K	GT	D	0.214	1.605	6.563	0.292	0.673	0.884	0.957
Eigen *et al* [17] Fine	K	GT	D	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Zhou *et al* [2]	K	GT	-	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Mahjourian *et al* [6]	K	GT	-	0.163	1.240	6.220	0.250	0.762	0.916	0.968
Zhichao *et al* [3]	K	GT	-	0.155	1.296	5.857	0.233	0.793	0.931	**0.973**
Ranjan *et al* [16]	K	GT	-	**0.148**	**1.149**	**5.464**	**0.226**	**0.815**	**0.935**	**0.973**
Pinard *et al* [11]	S	P	D + O	0.5071	7.1540	9.6209	0.5032	0.3960	0.6600	0.8138
Zhou *et al*	K	P	-	0.2786	**2.7059**	7.2956	0.3552	0.5816	0.8082	0.8982
Ours	K	P	-	0.3124	5.0302	8.4985	0.4095	0.5919	0.7961	0.8821
Ours	S → K	P	D+O → -	0.2940	3.9925	7.5727	0.3756	0.6092	0.8336	0.9090
Ours	K	P	O	0.2756	3.9335	**7.2939**	0.3539	0.6417	0.8457	0.9179
Ours	S → K	P	D+O → O	**0.2706**	4.4947	7.3119	**0.3452**	**0.6778**	**0.8564**	**0.9242**

Table 1. Quantitative tests on KITTI [1] Eigen split [17]. Measures are the same as in Eigen *et al* [17]. For blue measures, lower is better, for red measures, higher is better. For training, K is the KITTI dataset [1], S is the Still Box dataset [11]. For scale factor, GT is ground truth, P is pose. When scale was determined with pose, we discarded frames where GPS uncertainty was greater than $1m$. For supervision, D is depth and O is orientation. → denotes fine tuning.

Method	scale factor	supervision	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Pinard *et al* [11]	P	D + O	**0.2120**	**2.0644**	**7.0669**	**0.2959**	**0.7091**	**0.8810**	**0.9460**
Zhou *et al* [2]	GT	-	0.5005	11.4189	15.7207	0.6012	0.4969	0.6767	0.7671
Zhou *et al* [2]	P	-	0.8109	11.9956	17.2740	0.6928	0.3475	0.5733	0.7136
Ours	P	-	0.4684	10.9247	15.7560	0.5440	0.4524	0.6772	0.8037
Ours	P	O	**0.2970**	**5.2827**	**10.5090**	**0.4041**	**0.6684**	**0.8405**	**0.9058**

Table 2. Quantitative tests on StillBox, no pretraining has been done. The supervised Pinard *et al* [11] method is here to give an hint on a theoritical limit since it uses the same network, but with depth supervision

Table 1 presents quantitative results compared to prior works. We tried 5 different versions of our network. The first one is the exact same as [11], only trained on StillBox. It serves as a baseline purpose, without finetuning. The other four configurations are training from scratch or finetuning from StillBox, and training with orientation supervision or not.

As we might expect, on KITTI our method fails to converge as well as single image methods using classic relative depth quality measurement. However, when scale factor is determined from poses, we match the performance of the adapted

method from [2]. It can also be noted that finetuning provides a better starting point for our network, and that when available on a training set, orientation supervision is very advantageous.

When trying to train a Depth network with stabilized videos, it is then strongly recommended to do a first supervised training on a synthetic dataset such as StillBox.

Some failed test cases can be seen on Fig.4. The main sources of error are moving objects and poorly textured areas (especially concrete roads), even though we applied depth smooth geometric loss. Our attempt at explaining this failure is the large optical flow value compared to low textured area, meaning matching spatial structures is difficult. However, as KITTI acquisition rate is only 10 fps, we believe this problem would be less common on regular cameras, with typical rates of 30 fps or higher.

Table 2 presents results on the updated Still Box dataset. Zhou *et al* [2] performs surprisingly well given the theoretical lack of visual context in this dataset. However, our method performs better, whether from orientation supervision or completely unsupervised. We also compare it to supervised DepthNet from [11]. This can be considered a theoretical limit for training DepthNet on Still Box since [11] is completely supervised, and our training method is very close to it, indicating the good convergence of our model and thus our training algorithm and loss design validity.

5.5 Domain adaptation results

Finally, Fig.3 (left) compares Zhou *et al* [2] and our method on some test frames of our small UAV dataset. Our methods shows much better domain adaptation when fine tuning in a few-shot learning fashion. Especially for foreground objects, as Zhou *et al* [2] blends it with the trees near the horizon, which is very problematic for navigation. A video with result comparison is provided as supplementary material to this paper.

Table 3 compares domain robustness without any training: we tried inference on an upside-down KITTI test set, with ground up and sky down, and our method performs much better than Zhou *et al* [2], which is completely lost and performs worse than inferring a constant plane. However, our method is not as performing as Pinard *et al* [11] which score was expected to be the same as in Table 1, since it has not been trained on any KITTI frames, whether regular or reversed. This shows that our network may have learned to infer depth from both motion and context, which can be considered as a compromise between our two competitors, Zhou *et al* [2] relying only on context and Pinard *et al* [11] only on motion.

6 Conclusion and future work

We have presented a novel method for unsupervised depth learning, using not only depth from context but also from motion. This method leverages the context

Method	training set	scale factor	supervision	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Pinard *et al* [11]	S	P	D + O	**0.4622**	**6.0229**	**9.2277**	**0.4807**	**0.4149**	**0.6863**	**0.8349**
Constant Plane	-	GT	-	0.4568	4.8516	12.0848	0.6000	0.2962	0.5488	0.7524
Zhou *et al* [2]	K	GT	-	0.5931	7.5410	12.9943	0.7340	0.2223	0.4342	0.6263
Zhou *et al* [2]	K	P	-	1.5879	62.1068	21.1424	0.9579	0.1688	0.3260	0.4744
Ours	S → K	P	-	**0.6484**	**15.3906**	**12.4324**	0.6245	0.3820	0.6168	0.7607
Ours	S → K	P	O	0.7158	18.8145	12.5424	**0.5987**	**0.4024**	**0.6370**	**0.7723**

Table 3. Quantitative tests on upside-down KITTI [1]: sky is down and ground is up. No training has been done. Constant Plane outputs the same depth for every pixel

of stabilized videos, which is a midway between common use cases of stereo rigs and unconstrained ego-motion. As such, our algorithm provides a solution with embedded deployment in mind, especially for UAVs navigation, and requires only video and inertial data to be used.

Our method is also much more robust to domain changes, which is an important issue when dealing with deployment in large scale consumer electronics on which it is impossible to predict all possible contexts and situations, and our method outperforms single frame systems on unusual scenes.

The greatest limitation of our algorithm is the necessity of rigid scenes *even in inference*. Indeed, for single frame depth estimation or stereo, rigidity is not necessary during training. If an object moves between two frames, depth errors may compensate over the dataset if the object's movements are evenly distributed. As a consequence, some datasets, although usable for training, are not particularly suited for quality measurement since some scenes are not rigid. This is a strong limitation and one of our goals for a future work.

References

1. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11) (2013) 1231–1237
2. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: CVPR. (2017)
3. Yin, Z., Shi, J.: Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In: CVPR. (2018)
4. Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: CVPR. (2017)
5. Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., Fragkiadaki, K.: Sfm-net: Learning of structure and motion from video. CoRR **abs/1704.07804** (2017)
6. Mahjourian, R., Wicke, M., Angelova, A.: Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. CoRR **abs/1802.05522** (2018)
7. Saxena, A., Sun, M., Ng, A.Y.: Make3d: Learning 3d scene structure from a single still image. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(5) (May 2009) 824–840
8. Zbontar, J., LeCun, Y.: Stereo matching by training a convolutional neural network to compare image patches. Journal of Machine Learning Research **17** (2016) 1–32

9. Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A.: End-to-end learning of geometry and context for deep stereo regression. CoRR **abs/1703.04309** (2017)
10. N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, T.Brox: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). (2016) arXiv:1512.02134.
11. Pinard, C., Chevalley, L., Manzanera, A., Filliat, D.: End-to-end depth from motion with stabilized monocular videos. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences **IV-2/W3** (2017) 67–74
12. Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., Brox, T.: Demon: Depth and motion network for learning monocular stereo. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
13. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in neural information processing systems. (2015) 2017–2025
14. Garg, R., BG, V.K., Carneiro, G., Reid, I.: Unsupervised cnn for single view depth estimation: Geometry to the rescue. In: European Conference on Computer Vision, Springer (2016) 740–756
15. Xie, J., Girshick, R., Farhadi, A.: Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In: European Conference on Computer Vision, Springer (2016) 842–857
16. Ranjan, A., Jampani, V., Kim, K., Sun, D., Wulff, J., Black, M.J.: Adversarial collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. CoRR **abs/1805.09806** (2018)
17. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: Advances in neural information processing systems. (2014) 2366–2374
18. Pinard, C., Chevalley, L., Manzanera, A., Filliat, D.: Multi range Real-time depth inference from a monocular stabilized footage using a Fully Convolutional Neural Network. In: European Conference on Mobile Robotics, Paris, France, ENSTA ParisTech (September 2017)
19. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4) (2004) 600–612
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014)
21. Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., v.d. Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: IEEE International Conference on Computer Vision (ICCV). (2015)
22. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015)
23. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). (2010) 807–814
24. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: NIPS-W. (2017)
25. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)