Multi range Real-time depth inference from a monocular stabilized footage using a Fully Convolutional Neural Network

Clément Pinard[a,b] , Laure Chevalley[a], Antoine Manzanera[b], David Filliat[b]

***Abstract*— We propose a neural network architecture for depth map inference from monocular stabilized videos with application to UAV videos in rigid scenes. Training is based on a novel synthetic dataset for navigation that mimics aerial footage from gimbal stabilized monocular camera in rigid scenes.**

Based on this network, we propose a multi-range architecture for unconstrained UAV flight, leveraging flight data from sensors to make accurate depth maps for uncluttered outdoor environment.

We try our algorithm on both synthetic scenes and real UAV flight data. Quantitative results are given for synthetic scenes with a slightly noisy orientation, and show that our multi-range architecture improves depth inference.

Along with this article is a video that present our results more thoroughly.

I. Introduction

Scene understanding from vision is a core problem for autonomous vehicles and for UAVs in particular. In this paper we are specifically interested in computing the depth of each pixel from image sequences captured by a camera. We assume our camera's velocity (and thus displacement between two frames) is known, as most UAV flight systems include a speed estimator, allowing to settle the scale invariance ambiguity of the depth map.

Solving this problem could be beneficial for several problems such as environment scanning or applying depth-based sense and avoid algorithms for lightweight embedded systems that only have a monocular camera. Not relying on depth Sensors such as stereo vision, ToF camera, LiDar or Infra Red emitter/receiver allows to free the UAV from their weight, cost and limitations. Specifically, along with some RGB-D sensors being unable to operate under sunlight (e.g. IR and ToF), most of them suffer from range limitations and can be inefficient in case we need long-range information such as trajectory planning [7]. Unlike RGB-D sensors, depth from motion is flexible w.r.t. displacement and thus robust to high speeds or high distances as choosing among previous frames gives us a wide range of different displacements. For estimating such depth maps, we designed an end-to-end learning architecture, based on a synthetic dataset and a fully convolutional neural network that takes as input an image pair taken at different times. No preprocessing such as optical flow computation, nor visual odometry is applied to the input, while the depth is directly provided as an output.

[a]Parrot, Paris, France
(clement.pinard, laure.chevalley)@parrot.com
[b]U2IS, ENSTA ParisTech, Université Paris-Saclay, Palaiseau, France
(clement.pinard, antoine.manzanera, david.filliat)@ensta-paristech.fr

a) b) c)

Fig. 1. Camera stabilization can be done via a) mechanic gimbal or b) dynamic cropping from fish-eye camera, for drones or c) hand-held cameras

We created a dataset of image pairs with random translation movements, with no rotation, and a constant displacement magnitude applied during the whole training.

The assumption about videos without rotation appears realistic for two reasons:

- Hardware rotation compensation is mainly a solved problem, even for consumer products, with IMU-stabilized cameras on consumer drones or hand-held steady-cam (Fig 1).
- this movement is somewhat related to human vision and vestibulo-ocular reflex (VOR) [2]. Our eyes orientation is not induced by head rotation, our inner ear among other biological sensors allows us to compensate parasite rotation when looking at a particular direction.

Using the trained network, we propose an algorithm for real condition depth inference from a stabilized UAV. Displacement from sensors is used to compute real depth map, as it only differs from the synthetic constant displacement images by a scale factor. Our network output also allows us to *a posteriori* optimize the depth inference. By adjusting frame shift to get a displacement that would make the network get the same disparity distribution as during its training, we lower the depth error for next inference. For example, with large distances, ideal displacement between two frames is higher, and thus the shift is also higher. Moreover, we use multiple batch inference to compute multiple depth maps centered around a particular range, and fuse them to get a high precision for both close and far objects, no matter the distance, given a sufficient displacement from the UAV.

II. Related Work

Deep Learning and Convolutional Neural Networks have recently been widely used for numerous kinds of vision problem such as classification [13] and hand-written digits recognition [14].

Depth from vision is one of the problems studied with neural network, and has been addressed with a wide range

of training solution. Some datasets [6], [19] allow a neural network to learn end-to-end depth or disparity [15], [22], [4]. Reprojection error has also been used for unsupervised training for depth from a single image [20], [23] or for disparity between two frames of a stereo rig [12], [5].

Depth from a single image, although interesting, suffers from a major drawback which is overfitting. No motion is given to the network during inference, and the resulting depth is inferred from context, whereas they can be decorrelated. This technique can be sufficient for road driving context with an obvious road in front of the camera, but for a UAV flight usage, we may have to deal with very heterogeneous scenes. On the other hand, depth from a stereo pair is only implying a single lateral movement, and lacks a forward component to appear realistic for any aerial stabilized footage.

For depth from more complex movement from a monocular camera, current state of the art methods tend to use motion, and especially structure from motion, and most algorithm do not rely on deep learning [1], [17], [11]. Prior knowledge w.r.t. scene is used to infer a sparse depth map with its density usually growing over time. These techniques also called SLAM are typically used with unstructured movement (translation and rotation with varying magnitudes), produce very sparse point-cloud based 3D maps and require heavy calculation to keep track of the scene structure and align newly detected 3D points to the existing ones.

Our goal is to compute a dense depth map (where every point has a valid depth) using only two frames from the same camera, at different times, and without prior knowledge on the scene and movement, apart from the lack of rotation and the scale factor.

III. End-to-end learning of Depth Inference

Inspired by flow estimation and disparity (which is essentially magnitude of optical flow vectors), a problem to which exist a lot of very convincing methods [8], [10], we set up an end-to-end learning workflow, by training a neural network to explicitly predict the depth of every pixel in a scene, from an image pair with constant displacement value.

A. Still Box Dataset

We design our own synthetic dataset, using the rendering software *Blender*, to generate an arbitrary number of random rigid scenes, composed of basic 3d primitives (cubes, spheres, cones and tores) randomly textured from an image set scrapped from *Flickr* (see Fig 2).

These objects are randomly placed and sized in the scene, and walls are added at large distances as if the camera was inside a box (hence the name). The camera is moving at a fixed speed value, but to an uniformly distributed random direction, which is constant for each scene. It can be anything from forward/backward movement to lateral movement (which is then equivalent to stereo vision).

B. Dataset augmentation

In our dataset, we store data in 10 images long videos, with each frame paired with its ground truth depth. This allows us to set *a posteriori* distances distribution with a variable temporal shift between two frames. If we use a baseline shift of 3 frames, we can e.g. assume a depth three times as great as for two consecutive frames (shift of 1). In addition, we can also consider negative shift, which will only change displacement direction without changing speed value. This allows us, given a fixed dataset size, to get more evenly distributed depth values to learn, and also to de-correlate images from depth, preventing over-fitting during training, that would result in a scene recognition algorithm and would poorly perform on a validation set.

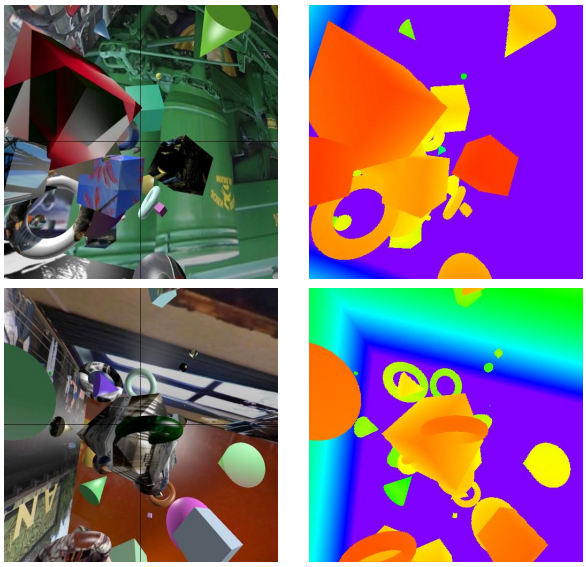

Fig. 2. Some examples of our renderings with associated depth maps (red is close, purple is far)

C. Depth Inference training

Our network is broadly inspired from FlowNetS [3] (initially used for flow inference) and called DepthNet. It is described in details in [18], we provide here a summary of its structure (Fig 3) and performances. Each convolution (apart from depth modules) is followed by a Spatial Batch Normalization and ReLU activation layer. Batch normalization helps convergence and stability during training by normalizing a convolution's output (0 mean and standard deviation of 1) over a batch of multiple inputs [9], and Rectified Linear Unit (ReLU) is the typical activation layer [21]. Depth Module are convolution modules, reducing the input to 1 feature map, which is expected to be the depth map, at a given scale. One should note that FlowNetS initially used LeakyReLU which has a non-null slope for negative values, but tests showed that ReLU performed better for our problem.

The main idea behind this network is that upsampled feature maps are concatenated with corresponding earlier convolution outputs (e.g. Conv2 output with Deconv5 output). Higher semantic information is then associated with information more closely linked to pixels (since it went through less downsampling convolutions) which is then used for reconstruction.

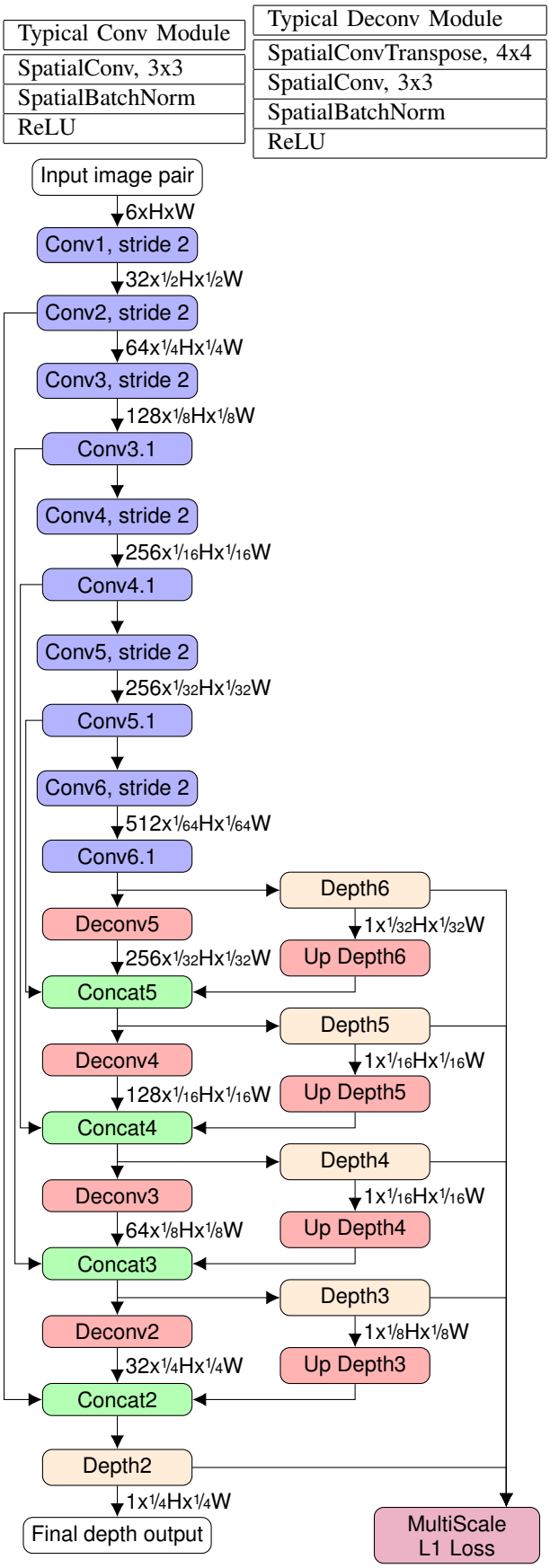

Fig. 3. DepthNet structure parameters, Conv and Deconv modules detailed above

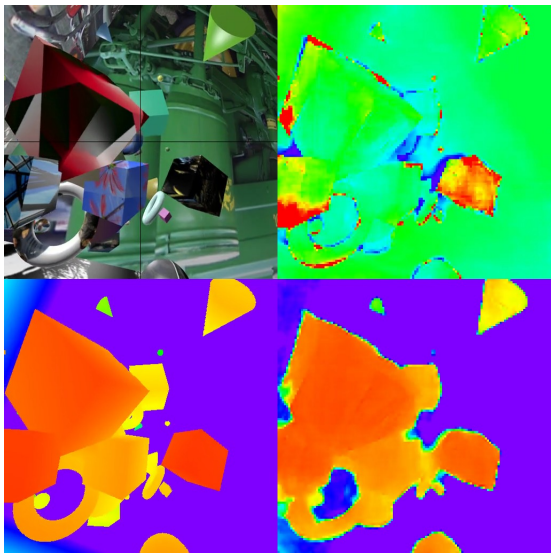

Fig. 4. Result on 512x512 images from $\text{DepthNet}_{64\to128\to256\to512}$. Upper-left: input, lower-left: Ground Truth depth, lower-right: our network output (128x128), upper-right: error, green is no error, red is overestimated depth, blue is underestimated

This multi-scale architecture has been proven very efficient for flow and disparity computing while keeping a very simple supervised learning process.

The main point of this experimentation is to show that direct depth estimation can be efficient regarding unknown translation. Like FlowNetS, we use a multi-scale criterion, with a L1 reconstruction error for each scale:

$$Loss = \sum_{s \in scales} \gamma_s \frac{1}{H_s W_s} \sum_i \sum_j |\beta_s(i,j) - \zeta_s(i,j)| \quad (1)$$

where

- γ_s is the weight of the scale, arbitrarily chosen.
- $(H_s, W_s) = (1/2^s H, 1/2^s W)$ are the height and width of the output.
- ζ_s is the scaled depth groundtruth, using average pooling.
- β_s is the ouput of the network at scale s.

As said earlier, we apply data augmentation to the dataset using different shifts, along with classic methods such a flips and rotations. We also clamp depth to a maximum of 100m, and provide sample pairs without shift, assuming its depth is 100m everywhere. As a consequence, the trained network will only be able to infer depth lower than 100m.

We applied training on several input size images, from 64x64 to 512x512. Fig 4 shows training results for mean L1 reconstruction error. Like FlowNetS, network output are downsampled by a factor of 4 with reference to the input size. As Table I shows, best results are obtained with multiple fine-tuning, with intermediate scales 64, 128, 256, and finally 512 pixels. Subscript values indicate finetuning processes. FlowNetS is performing better than DepthNet but by a fairly

Network	L1Error		RMSE	
	train	test	train	test
FlowNetS_{64}	1.69	4.16	4.25	7.97
DepthNet_{64}	2.26	4.49	5.55	8.44
$\text{FlowNetS}_{64\to128\to256\to512}$	0.658	**2.44**	1.99	**4.77**
$\text{DepthNet}_{64\to128}$	1.20	3.07	3.43	6.30
$\text{DepthNet}_{64\to128\to256}$	0.876	**2.44**	2.69	4.99
$\text{DepthNet}_{64\to128\to256\to512}$	1.09	2.48	2.86	**4.90**
$\text{DepthNet}_{64\to512}$	1.02	2.57	2.81	5.13
DepthNet_{512}	1.74	4.59	4.91	8.62

TABLE I. Quantitative results for depth inference networks. FlowNetS is modified with 1 channel outputs (instead of 2 for flow), trained from scratch for depth with Still Box, subscript indicates fine tuning process.

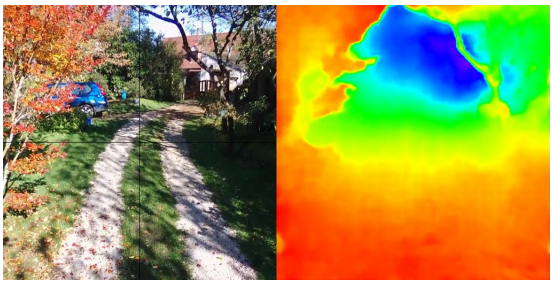

Fig. 5. Result on 512x512 real images input from a Bebop drone footage

light margin while being 5 times heavier and most of the time much slower.

IV. UAV NAVIGATION USE-CASE

A. Optimal frame shift determination

We learned depth inference from a moving camera, assuming its velocity is always the same. Results from real condition drone footage, on which we were careful to avoid camera rotation can be seen Fig 5. These results did not benefit from any fine-tuning from real footage, indicating that our Still Box Dataset, although not realistic in its scenes structures and rendering, appears to be sufficiently heterogeneous for learning to produce decent depth maps in real conditions. When running during flight, such a system can deduce the real depth map ζ from the network output and the drone displacement, knowing that the training displacement was D_0 (here $0.3m$)

$$\begin{aligned} \zeta(t) &= DepthNet(I_t, I_{t-\Delta t})\frac{D(t,\Delta t)}{D_0} \\ D(t,\Delta t) &= \left\| \int_{t-\Delta t}^{t} V(\tau) d\tau \right\| \end{aligned} \tag{2}$$

The actual correct interpretation of the output of DepthNet is rather a percentage than a distance. 100% meaning max distance for a given displacement D. We can introduce a function $\beta = \frac{DepthNet(I_t, I_{t-\Delta t})}{maxDistance}$ and a dimension-less parameter $\alpha = \frac{maxDistance}{D_0}$ for computing actual depth using the displacement D as the only distance related factor.

$$\zeta(t) = \alpha\beta(I_t, I_{t-\Delta t})D(t,\Delta t) \tag{3}$$

Depending of the depth distribution of the ground-truth depth map, it may be useful to adjust frame shift Δt. For example, when flying high above the ground with low speed, big structure detection and avoidance requires knowing precise distance values that are outside the typical range of any RGB-D sensor. The logical strategy would then be to increase the temporal shift between the frame pairs provided to DepthNet as inputs. More generally, one must provide inputs to DepthNet in order to ensure a well distributed depth output within its typical range. Depth-wise normalized error which is the essential quality measurement for values that we want to rescale, will diverge when ground truth depth approaches 0. Indeed, in addition to being equivalent to an infinite optical flow, the depth-wise error cannot tend to 0, which will make the expression ${}^{error}/_{depth}$ tend to $+\infty$ at 0 We thus need to choose the optimal spatial displacement and corresponding temporal shift to minimize error on the next inference, assuming the same depth distribution, to avoid too low or too high equivalent ground-truth. We chose the space displacement as:

$$D_{optimal}(t+1) = \frac{E(\zeta(t))}{\alpha\beta_{mean}} \tag{4}$$

With $E(\zeta(t))$ the mean of depth values and β_{mean} the optimal mean output of β, e.g. 0.5. $\Delta(t)$ is then computed numerically to get the frame shift with the closest corresponding displacement possible.

B. Multiple shifts inference

As neural network are traditionally computed within massively parallel architectures such as GPUs, multiple depth maps can be computed efficiently at the same time in a batch, especially for low resolution. Batch inference can then be used to compute depth with multiple shifts $\Delta(t,i)$. These multiple depth maps can then be combined to construct a higher quality depth map, with high precision for both long and short range. We propose a dynamic range algorithm, described Fig 6 to compute an combine different depth maps.

Instead of only one optimal displacement $D(t)$ from $E(\zeta)$, we use K-mean clustering algorithm [16] on the depth map to find a list of clusters on which each shift will focus. The clustering outputs a list of n centroids $C_i(\zeta)$ and corresponding $D_i(t)$ and $\Delta(t,i)$. n is an arbitrary chosen value, usually ranging from 1 to 4.

Final DepthMap is then computed from fusing these outputs using a weighted mean for each pixel. Each weight is actually a linear interpolation from 0 to 1 according to distance of depth from a target value β_{mean}. That way, fusion will favor values that are closer to this optimal value. An ϵ value is added to solve fusion when every depth map is off its wanted range.

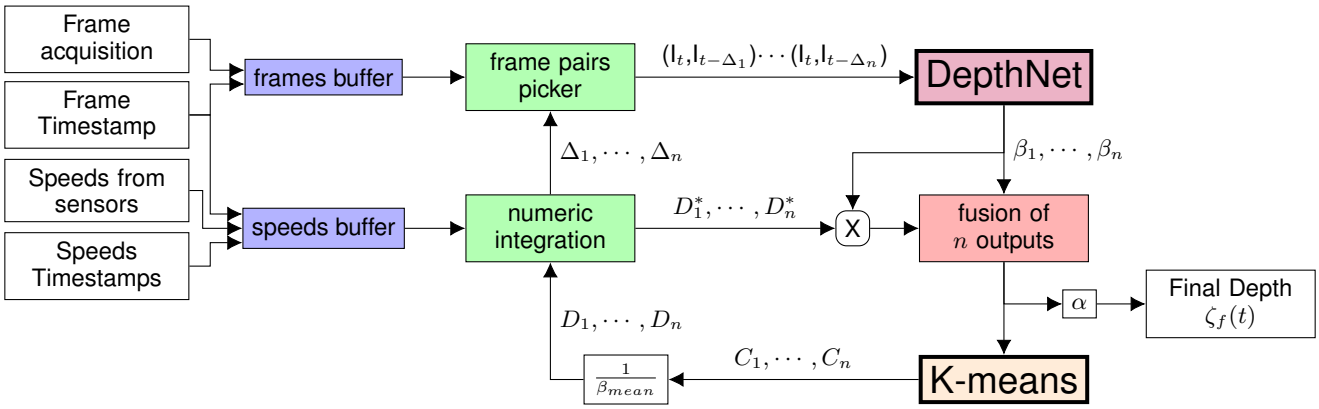

Fig. 6. Multiple shifts architecture. We used n different planes. Numeric integration, given a desired displacement D gives the closest possible displacement between frames D^* , along with corresponding shift Δ. As discussed in part IV, the fusion block computes pixel-wise weights from $\beta_1, \cdots, \beta_n$ to make a weighted mean of $\beta_1 D_1, \cdots, \beta_n D_n$

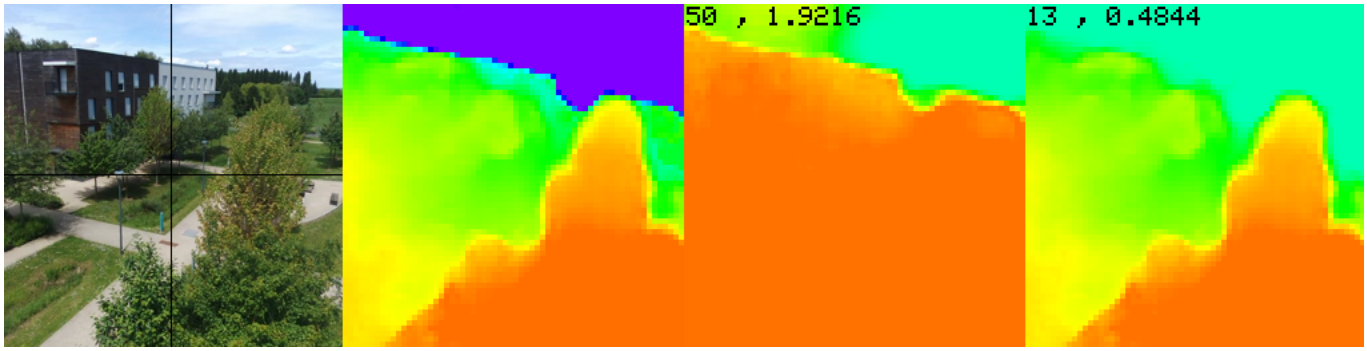

Fig. 7. real condition application of the multi-shift algorithm with Tiny DepthNet Clamped. First image is input. Last two are outputs of the network, for shifts of 50 and 13 with a drone flying forward at $1m.s^{-1}$ and at an altitude of $12m$, with corresponding displacements from sensors. Second is fused output, capped to $100m$ up

$$
\begin{aligned}
w_{ijk} &= \epsilon + f(\beta(img_t, img_{t-\Delta(t,i)})) \\
f : x \mapsto &\begin{cases} 0 & \text{if } x < \beta_{min} \\ \frac{x-\beta_{min}}{\beta_{mean}-\beta_{min}} & \text{if } \beta_{min} \leq x < \beta_{mean} \\ \frac{\beta_{max}-x}{\beta_{max}-\beta_{mean}} & \text{if } \beta_{mean} \leq x < \beta_{max} \\ 0 & \text{if } x \geq \beta_{max} \end{cases} \\
\zeta_i(t) &= \alpha D_i(t) \beta(img_t, img_{t-\Delta(t,i)})
\end{aligned} \tag{5}
$$

$$
\forall (j,k) \in [\![0, W]\!] \times [\![0, H]\!], \zeta_f(t)_{jk} = \frac{\sum_i w_{ijk} \zeta_{ijk}(t)}{\sum_i w_{ijk}} \tag{6}
$$

For our use-case, we set $\beta_{min} = 0.1$, $\beta_{mean} = 0.4$, $\beta_{max} = 0.9$ and $\epsilon = 10^{-3}$. i is the index of frame shift, j, k are the spatial indices. Fig 7 shows a result of the proposed algorithm for a batch size of 2. Notice how the high shift detects buildings while low shift detects trees.

C. Clamped DepthNet

Our proposed algorithm is actually suffering a problem for real condition videos, because we assume a perfect stabilization. Therefore, on very far objects (e.g. the sky), any minor optical flow caused by a default in stabilization will result in a massive error in depth. Moreover, our network being very good at recognizing shapes and giving it the same depth everywhere, this can result in the whole sky being computed as relatively close. We thus propose a network designed for a simpler problem: during training on still box, we clamp depth from $10m$ to $60m$, with a shift of 5 images (instead of 3 for DepthNet). These new parameters allow the network to only focus on mid range objects, dismissing close and far objects with respectively a too large and too small optical flow. This training workflow is very well suited for multiple shift depth inference. Every image pair will have a dedicated depth to analyze, allowing the fusion to not be bothered with redundant data, because of the high initial range of DepthNet.

Figure 8 shows results for multiples synthetic 256x256 scenes with ground truth, along with inference speed and a small noise added to camera initial orientation at each frame. $R(t) = R_0 + Euler(N_0 \mu(t))$, with $\mu(t)$ being a 3-dimensional random unit vector and N_0 a constant fixed to 0.001. We also report performance a thin version of our clamped network, that shows better results than DepthNet with 1 plane only in this noisy setup. The thin network has the same depth, but every convolution has an output half the number of feature maps of the original DepthNet. These

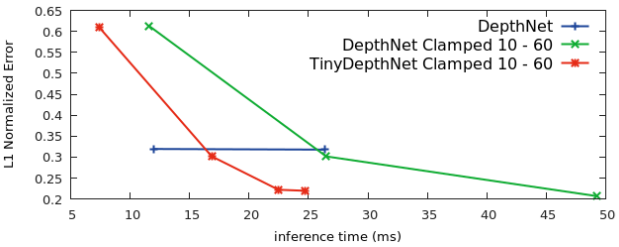

Fig. 8. results for synthetic 256x256 scenes with noisy orientation. DepthNet has been tested with 1 and 2 planes, DepthNet Clamped with 1 to 3 planes and Tiny DepthNet Clamped with 1 to 4 planes. Y axis is Absolute mean error (m) divided by ground-truth depth, X axis is inference speed, in ms

results have been obtained on a Quadro K2200m powered laptop.

V. Conclusion and Future Work

We proposed a novel way of computing dense depth maps from motion, along with a very comprehensive dataset for stabilized footage analysis and a technique for dynamic range real flight computing. This algorithm can then be used for depth-based sense and avoid algorithms in a very flexible way, in order to cover all kinds of path planning, from collision avoidance to long range obstacle bypassing.

A more thorough presentation of the results can be viewed in this video. `http://perso.ensta-paristech.fr/~manzaner/Download/ECMR2017/DepthNetResults.mp4`

Future works include implementation of such a path planning algorithm, and construction of a real condition fine tuning dataset, using UAVs footages and a preliminary thorough 3D offline scan. This would allow us to measure quantitative quality of our network for real footages and not only subjective as for now. We could also use unsupervised techniques, using re-projection errors as in [23].

We also believe that our network can be extended to re-inforcement learning applications that will potentially result in a complete end-to-end sense and avoid neural network for monocular cameras.

The major drawback of our algorithm is however the necessity for a scene to be rigid. This is obviously never the case, and even though UAV footage are less prone to moving objects like in autonomous driving problems, we will have this issue whenever a moving target is to be followed. To solve this problem, an explicit movement equation for both the camera and the moving targets may have to be computed, as in [20]. In any case, this problem will be a challenge and may not be solvable with fully Convolutional networks only as we did in this article.

References

[1] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[2] Lorente De Nó, R. Vestibulo-ocular reflex arc. *Archives of Neurology & Psychiatry*, 30(2):245–291, 1933.

[3] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[4] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[5] Ravi Garg, Vijay Kumar B. G, and Ian D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. *CoRR*, abs/1603.04992, 2016.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[7] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[8] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *arXiv preprint arXiv:1612.01925*, 2016.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-End Learning of Geometry and Context for Deep Stereo Regression. *ArXiv e-prints*, March 2017.

[11] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[12] Kishore Reddy Konda and Roland Memisevic. Unsupervised learning of depth and motion. *CoRR*, abs/1312.3429, 2013.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016.

[16] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[17] Raul Mur-Artal and Juan D Tardos. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475*, 2016.

[18] Clément Pinard, Laure Chevalley, Antoine Manzanera, and David Filliat. End-to-end depth from motion with stabilized monocular videos. In *submitted to the International Conference on Unmanned Aerial Vehicles in Geomatics (UAV-G)*, 2017.

[19] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[20] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. SfM-Net: Learning of Structure and Motion from Video. *ArXiv e-prints*, April 2017.

[21] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[22] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599, 2015.

[23] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.