End-to-end depth from motion with stabilized monocular videos
Clément Pinard, Laure Chevalley, Antoine Manzanera, David Filliat

To cite this version:

HAL Id: hal-01587652
https://hal.archives-ouvertes.fr/hal-01587652
Submitted on 14 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
END-TO-END DEPTH FROM MOTION WITH STABILIZED MONOCULAR VIDEOS

C. Pinard\textsuperscript{a,b}, L. Chevalley\textsuperscript{a}, A. Manzanera\textsuperscript{a}, D. Filliat\textsuperscript{b}

\textsuperscript{a}Parrot, Paris, France - (clement.pinard, laure.chevalley)@parrot.com
\textsuperscript{b}U2IS, ENSTA ParisTech, Universite Paris-Saclay, Palaiseau, France - (antoine.manzanera, david.filliat)@ensta-paristech.fr

KEY WORDS: Dataset, Navigation, Monocular, Depth from Motion, End-to-end, Deep Learning

ABSTRACT:

We propose a depth map inference system from monocular videos based on a novel dataset for navigation that mimics aerial footage from gimbal stabilized monocular camera in rigid scenes. Unlike most navigation datasets, the lack of rotation implies an easier structure from motion problem which can be leveraged for different kinds of tasks such as depth inference and obstacle avoidance. We also propose an architecture for end-to-end depth inference with a fully convolutional network. Results show that although tied to camera inner parameters, the problem is locally solvable and leads to good quality depth prediction.

1. 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 a pair of consecutive images 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.

Solving this problem could be beneficial for applying depth-based sense and avoid algorithms for lightweight embedded systems that only have a monocular camera and cannot directly provide a RGB-D image. This could allow such devices to go without heavy or power expensive dedicated devices such as ToF camera, LiDar or Infra Red emitter/receiver (Hitomi et al., 2015) that would greatly lower autonomy. In addition, many RGBD sensors are unable to operate under sunlight (e.g. IR and ToF), and most of them suffer from range limitations and can be inefficient in case we need long-range trajectory planning (Hadsell et al., 2009). The faster an UAV is, the longer the range we will need to efficiently avoid obstacles. Unlike RGB-D sensors, depth from motion is robust to high speeds since it will be normalized by the displacement between two frames. Given the difficulty of the task, several learning approaches have been proposed to solve it.

A large number of datasets has been developed in order to propose supervised learning and validation for fundamental vision tasks, such as optical flow (Geiger et al., 2012, Dosovitskiy et al., 2015, Weinzaepfel et al., 2013) stereo disparity and even 3D scene flow (Menze and Geiger, 2015, N.Mayer et al., 2016). These different measures can help figure up scene structure and camera motion, but they remain low-level in terms of abstraction. End-to-end learning of a certain value such as three dimensional geometry may be hard to compute on a totally unrestricted monocular camera movement.

We focus on RGB-D datasets that would allow supervised learning of depth. RGB pairs (preferably with the corresponding displacement) being the input, and D the desired output. It turns out that today, for learning depth from video, the choice among existing RGB-D datasets is either unrestricted w.r.t. ego-motion

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
2. RELATED WORK

2.1 Monocular vision based sense and avoid

Sense and avoid problems are mostly approached using a dedicated sensor for 3D analysis. However, some works have been done trying to leverage Optical flow from Monocular camera (Sohilah and Karim, 2007, Zingg et al., 2010). They highlight the difficulty of estimating depth solely with flow, especially when the camera is pointed toward movement. One can note that rotation compensation was already used with fish-eye camera in order to have a more direct link between flow and depth. Another work (Coombs et al., 1998) also demonstrated that basic obstacle avoidance could be achieved in cluttered environments such as a closed room.

Some interesting work concerning obstacle avoidance from Monocular camera (LeCun et al., 2005, Hadsell et al., 2009, Michels et al., 2005) showed that single frame analysis can be more efficient than depth from stereo for path planning. However, these works were not applied on UAV, on which depth cannot be directly deduced from distance to horizon, because obstacles and paths are three-dimensional.

More recently, Giusti et al. (Giusti et al., 2016) showed that a monocular system can be trained to follow a hiking path. But once again, only 2D movement is approached, asking a UAV going forward to change its yaw based on likeliness to be following a traced path.

2.2 Depth inference

Deep Learning and Convolutional Neural Networks have recently been widely used for numerous kinds of vision problem such as classification (Krizhevsky et al., 2012) and have also been used to generate image with meaningful content (Long et al., 2015)

Depth from vision is one of the problems studied with neural network, and has been addressed not only with image pairs, but also single images (Eigen et al., 2014, Saxena et al., 2005, Garg et al., 2016, Godard et al., 2017). Depth inference from stereo has also been widely studied (Luo et al., 2016, Zbontar and LeCun, 2015).

Current state of the art methods for depth from monocular view tends to use motion, and especially structure from motion, and most algorithms do not rely on deep learning (Cadena et al., 2016, Mur-Artal and Tardos, 2016, Klein and Murray, 2007, Pizzoli et al., 2014). 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, 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. However SLAM is generally used for off-line 3D scan rather than obstacle avoidance.

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

2.3 Navigation datasets

As discussed earlier, numerous datasets exist with depth ground truth, but to our knowledge, no dataset proposes only translational movement. Some provide IMU data along with frames (Smith et al., 2009, Gaidon et al., 2016), that could be used to compensate rotation but their size in terms of different scenes would only let us use them for finetuning or validation.

3. STILL BOX DATASET

From our understanding of the lack of an adequate dataset for our problem, we decided to design our own synthetic dataset.

We used 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, so that they are mostly in front of the camera, with possible variations including objects behind camera, or even camera inside an object.

Scenes in which camera goes through objects are discarded. To add difficulty we also applied uniform textures on a proportion of the primitives. Each primitive thus has a uniform probability (corresponding to texture ratio) of being textured from a color ramp and not from a photograph.

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, to a uniformly distributed random direction, which remains constant for each sequence. It can be anything from forward/backward movement to lateral movement (which is then equivalent to stereo vision). Tables 1 and 2 show a summary of our scenes parameters. They can be changed at will, and are stored in a metadata JSON file to keep track of it. Our dataset is then composed of 4 sub-datasets with different resolutions, 64px dataset being the largest in terms of number of samples, 512px being the heaviest in data.

4. END-TO-END LEARNING OF DEPTH INFERENCE

4.1 Why not disparity?

Flow Estimation and disparity (which is essentially magnitude of optical flow vectors) are problems for which there is a lot of
very convincing methods (Ilg et al., 2016, Kendall et al., 2017).

Knowing depth and displacement in our dataset, one could think it is easy to get disparity, train a network for it using existing optical flow methods, and get the depth map indirectly. The link between disparity and depth is actually prone to large errors. To understand why, we can consider a picture with \((u, v)\) coordinates, and optical center at \(P_0 = (u_0, v_0)\), and define our system as follows:

**Definition 1** Disparity \(\delta(P)\) is defined here by the norm of the velocity flow, \(\text{flow}(P) = \left(\frac{du}{dv}\right)\) of a point \(P = (u, v)\).

\[
\forall P = (u, v), \delta(P) = \|\text{flow}(P)\|
\]

**Definition 2** Focus of Expansion is defined by the point \(\Phi\) where each flow vector \(\text{flow}(P) = \left(\frac{du}{dv}\right)\) of a point \(P = (u, v)\) is headed from. Note that this property is only true in a rigid scene and without rotation. One can note that for a pure translation, \(\Phi\) is the projection of the displacement vector of the camera onto its own focal plane.

\[
\forall P = (u, v), \text{det}\left(\begin{bmatrix} u & \frac{du}{dv} \\ v & \frac{dv}{du} \end{bmatrix} \right) = 0
\]

**Theorem 1** For a random rotation-less displacement of norm \(V\) of a pinhole camera, with a focal length \(f\), depth \(\zeta(P)\) is an explicit function of disparity \(\delta(P)\), focus of expansion \(\Phi\) and optical center \(P_0\).

\[
\forall P, \zeta(P) = \frac{V f}{\sqrt{f^2 + \|P_0\Phi\|^2}} \left(\frac{\|P_0\Phi\|}{\delta(P)} - 1\right)
\]

This result (proved in annex A) is in a useful form for limit values. Lateral movement corresponds to \(\|P_0\Phi\| \to +\infty\) and then

\[
\lim_{\|P_0\Phi\| \to +\infty} \zeta(P) = \frac{f V}{\delta(P)}
\]

When approaching \(\Phi\), knowing that depth is a bounded positive value, we can deduce:

\[
\delta(P) \to \frac{\|P_0\Phi\|}{\delta(P)}\]

Limit of disparity in this case is 0 and we use its inverse. As a consequence, small errors on disparity estimation will result in diverging values of depth near the focus of expansion, while it corresponds to the direction the camera is moving to, which is clearly problematic for depth-based obstacle avoidance.

Given the random direction of our camera’s displacement, computing depth from disparity is therefore much harder than for a classic stereo rig. To tackle this problem, we decided to set up an end-to-end learning workflow, by training a neural network to explicitly predict the depth of every pixel in the scene, from an image pair with constant displacement value \(D_0\).

### 4.2 Dataset augmentation

The way we store data in 10 images long videos, with each frame paired with its ground truth depth, allows us to set a posteriori distances distribution with a variable temporal shift between two frames. For example, using a baseline shift of 3 frames, we can assume a depth three times greater than for two consecutive frames (shift of 1). A shift of 0 will result in an infinite depth, capped to a certain value (here 100m). In addition, we can also consider negative shift, which will only change displacement direction without changing speed value. These augmentation techniques allow us, given a fixed dataset size, to get more evenly distributed depth values to learn, and also to decorrelate appearance from depth, preventing any over-fitting during training, that would result in a scene recognition algorithm and would perform poorly on a validation set.

### 4.3 Depth Inference training

Our network is broadly inspired from FlowNetS (Dosovitskiy et al., 2015) (initially used for flow inference) and called DepthNet. It is described Fig 3. Every convolution (apart from depth module) 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 over a batch of multiple inputs (Ioffe and Szegedy, 2015), and Rectified Linear Unit (ReLU) is the typical activation layer (Xu et al., 2015). 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. Higher semantic information is then associated with information more closely linked to pixels (since it went through less strided convolutions) which is then used for reconstruction. It is also important to note that contrary to FLowNetS, our network does not need to be completely agnostic to optical flow structure, as it will be given frame pairs from rigid scenes, which dramatically lowers our problem’s dimensionality. We therefore assumed that a thinner network would perform as well as a simple variation of FlowNetS and used half the feature maps for each layer.

This multi-scale architecture proved to be very efficient for flow and disparity computing while keeping a very simple supervised learning process. Our network is admittedly very simple and one could leverage some more advanced work that were used for flow
Figure 3. DepthNet structure parameters, Conv and Deconv modules detailed above.

Figure 4. result for 64x64 images, upper-left: input (before being downsampled to 64x64), lower-left: Ground Truth depth, lower-right: our network output (16x16), upper-right: error (green is no error, red is over-estimated depth, blue is under-estimated) and disparity, such as FlowNetC or GC-Net (Kendall et al., 2017) among many others. The main point of this experimentation is to show that direct depth estimation can be beneficial regarding unknown translation. Like FlowNetS, we use a multi-scale criterion, with a L1 reconstruction error for each scale.

\[
Loss = \sum_{s\in\text{scales}} \frac{1}{H_s W_s} \sum_i \sum_j |\text{output}_s(i,j) - \zeta_s(i,j)|
\]

where

- \(\gamma_s\) is the weight of the scale, arbitrarily set to \(W_s\) in our experiments.
- \((H_s, W_s) = (\frac{1}{2^n} H, \frac{1}{2^n} W)\) are the height and width of the output.
- \(\zeta_s\) is the scaled depth groundtruth, using average pooling.

As said earlier, we apply data augmentation to the dataset using different shifts, along with classic methods such as flips and rotations.

Fig 4 shows results from 64x64 dataset. Like FlowNetS, results are downsampled by a factor of 4, which gives 16x16 Depth Maps.

One can notice that although the network is still fully convolutional, feature map sizes go down to 1x1 and then behave exactly like a Fully Connected Layer, which can serve to implicitly figure out motion direction, and spread this information across the outputs. The second noticeable fact is that near FOE, (see Fig 5 for pixel-wise depth error according to distance from FOE) the network has no difficulty in inferring depth, which means that it uses neighbor disparity and interpolates when no other information is available.

This can be interpreted as 3d shapes identification, along with their magnification: pixels belonging to the same shape are deemed to have close and continuous depth values, resulting in a FOE-independent depth inference.
Figure 5. mean depth error as a function of distance from FOE, for DepthNet

<table>
<thead>
<tr>
<th>Network</th>
<th>L1Error (train)</th>
<th>L1Error (test)</th>
<th>RMSE (train)</th>
<th>RMSE (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlowNetS32→128→256→512</td>
<td>0.658</td>
<td>2.44</td>
<td>1.99</td>
<td>4.77</td>
</tr>
<tr>
<td>FlowNetS64</td>
<td>2.26</td>
<td>4.49</td>
<td>5.55</td>
<td>8.14</td>
</tr>
<tr>
<td>DepthNet64→128</td>
<td>1.20</td>
<td>3.07</td>
<td>3.43</td>
<td>6.30</td>
</tr>
<tr>
<td>DepthNet64→128→256</td>
<td>0.870</td>
<td>2.44</td>
<td>2.69</td>
<td>4.99</td>
</tr>
<tr>
<td>DepthNet64→128→256→512</td>
<td>1.09</td>
<td>2.48</td>
<td>2.80</td>
<td>4.90</td>
</tr>
<tr>
<td>DepthNet128</td>
<td>1.02</td>
<td>2.57</td>
<td>2.81</td>
<td>5.13</td>
</tr>
<tr>
<td>DepthNet512</td>
<td>1.74</td>
<td>4.59</td>
<td>4.91</td>
<td>8.62</td>
</tr>
</tbody>
</table>

Table 3. Quantitative results for depth inference networks. FlowNetS is modified with 1 channel outputs (instead of 2 for inferring flow), and trained from scratch for depth with Still Box. Subscript indicates fine tuning process.

4.4 From 64px to 512px Depth inference

One could think that a fully convolutional network such as ours can not solve depth extraction for pictures greater than 64x64. The main idea is that for a fully convolutional network, each pixel is applied the same operation. For disparity, this makes sense because the problem is essentially similarity from different picture shifts. However, we are on the picture, the operation is the same. For depth inference when FOE is not diverging (forward movement is non negligible), Theorem 1 shows that once you know the FOE, distinct operations should be done depending on the distance from the FOE and from the optical center $P_0$. The only possible strategy for a fully convolutional network would be to also compute the position in the image and to apply the compensating scaling to the output.

This problem then seems very difficult, if not impossible for a network as simple as ours, and if we run the training directly on

<table>
<thead>
<tr>
<th>Network</th>
<th>size</th>
<th>980Ti</th>
<th>Quadro K2200m</th>
<th>TX1</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlowNetS64</td>
<td>39.4</td>
<td>225</td>
<td>153</td>
<td>76</td>
</tr>
<tr>
<td>DepthNet64</td>
<td>7.33</td>
<td>364</td>
<td>245</td>
<td>190</td>
</tr>
<tr>
<td>FlowNetS128</td>
<td>39.4</td>
<td>69</td>
<td>8.8</td>
<td>16</td>
</tr>
<tr>
<td>DepthNet128</td>
<td>7.33</td>
<td>294</td>
<td>118</td>
<td>171</td>
</tr>
<tr>
<td>DepthNet256</td>
<td>39.4</td>
<td>178</td>
<td>36</td>
<td>121</td>
</tr>
<tr>
<td>DepthNet512</td>
<td>7.33</td>
<td>68</td>
<td>8.8</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 4. Size (millions of parameters) and Inference speeds (fps) on different devices. Batch sizes are 1 and 8 (when applicable). A batch size of 8 means 8 depth maps computed at the same time.
One of the drawbacks of this learning method is that the \( f \) value (which is focal length divided by pixel size) of our camera must be the same as the one used in training. Our dataset creation framework however allows us to change this value very easily for training. One must also be sure to have pinhole equivalent frames like during training.

Depending of the depth distribution of the groundtruth depth map, it may be useful to adjust frame shift. For example, when flying high above the ground, 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 ensure a well distributed depth map from 0 to 100m to get high quality depth inference. This problem can be solved by deducing optimal shift \( \Delta t \) from precedent inference distribution, e.g:

\[
\Delta t_{i+1} = \frac{E_c}{E_0}
\]

where \( E_0 \) is an arbitrarily chosen value around which our network has the best accuracy, e.g. 50m, because our network outputs from 0 to 100m, and \( E_c \) is the mean of precedent output, i.e.:

\[
E_c = \frac{1}{HW} \sum_{i,j} \text{DepthNet}(I_i, I_{i-\Delta t}), i,j
\]

6. 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. This algorithm can be used for depth-based sense and avoid algorithm in a very flexible way, in order to cover all kinds of path planning, from collision avoidance to long range obstacle bypassing.

Future works include using batch inference to compute depth with multiple shifts \( \Delta t \). As shown in Table 4, batch size greater than 1 can be used to some extent (especially for low resolution) to efficiently compute multiple depth maps.

\[
\zeta(t) = \text{DepthNet}(I_t, I_{t-\Delta t})
\]

These multiple depth maps can then be either combined to construct a high quality depth map, or used separately to run two different obstacle avoidance algorithm, e.g. one dedicated for long range path planning (and then a high value \( \Delta t \)) and the other for reactive and short range collision avoidance (with low \( \Delta t \)). While one depth map will display closer areas at zero distance but further regions with precision, the other will set far regions to infinity (or 100m for DepthNet) but closer region with high resolution.

We also consider an implementation of a real dataset for fine tuning, using UAVs footage and either a preliminary thorough 3D offline scan or groundtruth-less techniques (Zhou et al., 2017). This would allow us to measure quantitative quality of our network for real footages and not only subjective as for now.

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


Godard, C., Mac Aodha, O. and Brostow, G. J., 2017. Unsupervised monocular depth estimation with left-right consistency. In: CVPR.


relative movement of $P_{XYZ}$ is $-m$

so we have

$$
\begin{align*}
    dX &= -m_x \\
    dY &= -m_y \\
    dZ &= -m_z
\end{align*}
$$

If we compute $u_A$ for $P_A$:

$$
\begin{align*}
    u_A &= u_0 + f \frac{X - dX}{Z - dZ} \\
    du &= u_B - u_A = f \left( \frac{X}{Z} + \frac{X + m_x}{Z + m_x} \right) \\
    du &= \frac{m_x}{Z + m_x} \left( u_B - \Phi_u \right)
\end{align*}
$$

Similarly, with $v$, we get:

$$
\begin{align*}
    dv &= \frac{m_y}{Z + m_x} \left( u_B - \Phi_v \right)
\end{align*}
$$

We consider disparity $\delta(P)$ norm of the flow $\{du, dv\}$ expressed in frame B (which is correlated to depth at this frame).

$$
\begin{align*}
    \forall P = \begin{pmatrix} u \\ v \end{pmatrix}, \delta(P) = \left\| \frac{Pf}{\phi(P)} \right\| - 1
\end{align*}
$$

Consequently, we can deduce depth at frame B from disparity:

$$
\begin{align*}
    \forall P = \begin{pmatrix} u \\ v \end{pmatrix}, \delta(P) = \left\| \frac{Pf}{\phi(P)} \right\| - 1
\end{align*}
$$

From our dataset construction, we know that $m_x^2 + m_y^2 + m_z^2 = V^2$

Let us call

$$
\begin{align*}
    \Delta u &= u_0 - \Phi_u \\
    \Delta v &= v_0 - \Phi_v
\end{align*}
$$

From 3, we get:

$$
\begin{align*}
    m_x &= \frac{m_x \Delta u}{f} \\
    m_y &= \frac{m_y \Delta v}{f} \\
    V^2 &= m_z^2 \left( 1 + \frac{\Delta u^2 + \Delta v^2}{f^2} \right)
\end{align*}
$$

and then from 6 we get

$$
\begin{align*}
    \forall P, \delta(P) = \frac{Vf}{\sqrt{f^2 + \frac{m_z^2 f^2}{\phi(P)^2}}} \left( \frac{\phi(P)}{\phi(P) - 1} \right) \square
\end{align*}
$$

Let $m = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix}$ be $\overline{AB}$

$$
\Phi = \begin{pmatrix} \Phi_u \\ \Phi_v \end{pmatrix} = \begin{pmatrix} u_0 + \frac{m_x}{m_z} \\ v_0 + \frac{m_x}{m_z} \end{pmatrix}
$$

let $P_{XYZ}$ be a point $\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$. For camera B we have $P_B = \begin{pmatrix} u_B \\ v_B \end{pmatrix} = \begin{pmatrix} u_0 + \frac{X}{Z} \\ v_0 + \frac{X}{Z} \end{pmatrix}$