CodeSLAM — Learning a Compact, Optimisable Representation for Dense Visual SLAM

Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, Andrew J. Davison
Dyson Robotics Laboratory at Imperial College, Department of Computing, Imperial College London, UK
m.bloesch@imperial.ac.uk

Abstract

The representation of geometry in real-time 3D perception systems continues to be a critical research issue. Dense maps capture complete surface shape and can be augmented with semantic labels, but their high dimensionality makes them computationally costly to store and process, and unsuitable for rigorous probabilistic inference. Sparse feature-based representations avoid these problems, but capture only partial scene information and are mainly useful for localisation only.

We present a new compact but dense representation of scene geometry which is conditioned on the intensity data from a single image and generated from a code consisting of a small number of parameters. We are inspired by work both on learned depth from images, and auto-encoders. Our approach is suitable for use in a keyframe-based monocular dense SLAM system: While each keyframe with a code can produce a depth map, the code can be optimised efficiently jointly *with pose variables and together with the codes of overlapping keyframes to attain global consistency. Conditioning the depth map on the image allows the code to only represent aspects of the local geometry which cannot directly be predicted from the image. We explain how to learn our code representation, and demonstrate its advantageous properties in monocular SLAM.*

1. Introduction

The underlying representation of scene geometry is a crucial element of any localisation and mapping algorithm. Not only does it influence the type of geometric qualities that can be mapped, but also dictates what algorithms can be applied. In SLAM in general, but especially in monocular vision, where scene geometry cannot be retrieved from a single view, the representation of geometrical uncertainties is essential. However, uncertainty propagation quickly becomes intractable for large degrees of freedom. This difficulty has split mainstream SLAM approaches into two categories: *sparse* SLAM [5, 16, 20] which represents geometry by a sparse set of features and thereby allows joint probabilistic inference of structure and motion (which is a key pillar of probabilistic SLAM [6]) and *dense* or semi-dense SLAM [21, 9] that attempts to retrieve a more complete description of the environment at the cost of approximations to the inference methods (often discarding cross-correlation of the estimated quantities and relying on alternating optimisation of pose and map [22, 8]).

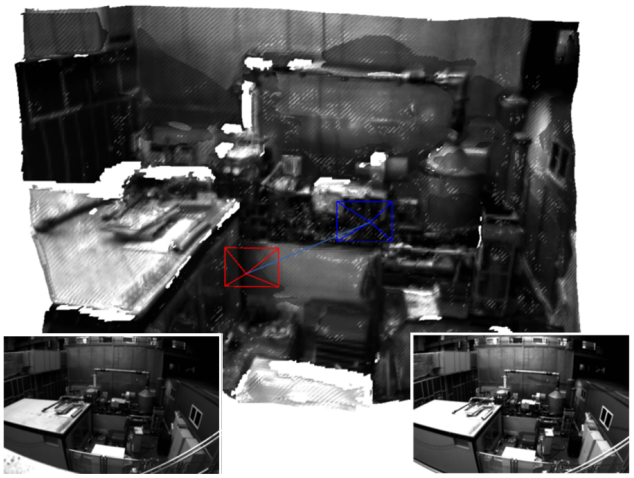

Figure 1. Two view reconstruction on selected frames from the EuRoC dataset. The proposed compact representation of 3D geometry enables joint optimisation of the scene structure and relative camera motion without explicit priors and in near real-time performance.

However, the conclusion that a dense representation of the environment requires a large number of parameters is not necessarily correct. The geometry of natural scenes is not a random collection of occupied and unoccupied space but exhibits a high degree of order. In a depth map, the values of neighbouring pixels are highly correlated and can often be accurately represented by well known geometric smoothness primitives. But more strongly, if a higher level

of understanding is available, a scene could be decomposed into a set of semantic objects (e.g. a chair) together with some internal parameters (e.g. size of chair, number of legs) and a pose, following a direction indicated by the SLAM++ system [26] towards representation with very few parameters. Other more general scene elements which exhibit simple regularity such as planes can be recognised and efficiently parametrised within SLAM systems (e.g. [25, 11]). However, such human-designed dense abstractions are limited in the fraction of natural, cluttered scenes which they can represent.

In this work we aim at a more generic compact representation of dense scene geometry by training an auto-encoder on depth images. While a straightforward auto-encoder might over-simplify the reconstruction of natural scenes, our key novelty is to condition the training on intensity images. Our approach is planned to fit within the common and highly scalable keyframe-based SLAM paradigm [16, 9], where a scene map consists of a set of selected and estimated historical camera poses together with the corresponding captured images and supplementary local information such as depth estimates. The intensity images are usually required for additional tasks, such as descriptor matching for place recognition or visualisation, and are thus readily available for supporting the depth encoding.

The depth map estimate for a keyframe thus becomes a function of the corresponding intensity image and an unknown compact representation (henceforth referred to as 'code'). This allows for a compact representation of depth without sacrificing reconstruction detail. In inference algorithms the code can be used as dense representation of the geometry and, due to its limited size, this allows for full joint estimation of both camera poses and dense depth maps for multiple overlapping keyframes. We might think of the image providing local details and the code as supplying more global shape parameters which are often not predicted well by 'depth from single image' learning. Importantly though, these global shape parameters are not a designed geometric warp but have a learned space which tends to relate to semantic entities in the scene, and could be seen as a step towards enabling optimisation in general semantic space.

Our work comes at a time when many authors are combining techniques from deep learning with estimation-based SLAM frameworks, and there is an enormously fertile field of possibilities for this. Some particularly eye-catching pieces of work over the past year have focused on supervised and self-supervised training of surprisingly capable networks which are able to estimate visual odometry, depth and other quantities from video [10, 29, 30, 4, 32, 31]. These methods run with pure feed forward network operation at runtime, but rely on geometric and photometric formulation and understanding at training time to correctly formulate the loss functions which connect different network components. Other systems are looking towards making consistent long-term maps by constantly refining geometric estimates, and this is the domain in which we are more interested here. In CNN-SLAM [28] single image depth prediction and dense alignment are used to produce a dense 3D map and this gives a promising result, but it is not possible to optimise the predicted depth maps further for consistency when multiple keyframes overlap as it is in our approach.

To summarise, the two key contributions of our paper are:

- The derivation of a compact and optimisable representation of dense geometry by conditioning a depth auto-encoder on intensity images.
- The implementation of the first real-time targeted monocular system that achieves such a tight joint optimisation of motion and dense geometry.

In the rest of this paper, we will first explain our method for depth learning and prediction, and then show the applicability of this approach in a SLAM setting.

2. Intensity Conditioned Depth Auto-Encoding

Two important qualities of geometry representations are *accuracy* and *practicality*. While the *accuracy* of a representation simply relates to its ability to reproduce the geometry, the *practicality* describes how well the representation can be used in an overall system. For inference-based SLAM systems, the latter typically requires the representation to lead to an optimisable loss function. For a representation G of the geometry a loss function $L(G)$ should be differentiable and have a clear minimum. Additionally, the size of the representation G should be limited in order to allow the estimation of second-order statistical moments (a covariance matrix) as part of more powerful inference methods.

In order to come up with a compact representation of the scene geometry we explore auto-encoder-like network architectures. Auto-encoders are networks which attempt to learn an identity mapping while being subject to an information bottleneck which forces the network to find a compact representation of the data [24]. In a naive attempt to auto-encode depth this would lead to very blurry depth reconstruction since only the major traits of the depth image can make it through the bottleneck (see Figure 2). In a monocular vision setup, however, we have access to the intensity images, which we are very likely to store alongside every keyframe. This can be leveraged to make the encoding more efficient: We do not need to encode the full depth information, but only need to retain the part of the information which cannot be retrieved from the intensities. The depth D

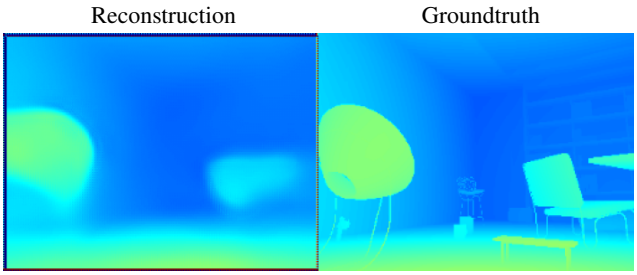

Figure 2. Depth auto-encoder without the use of image intensity data. Due to the bottleneck of the auto-encoder only major traits of the depth image can be captured.

thus becomes a function of image I and (unknown) code $\boldsymbol{c}$:

$$D = D(I, \boldsymbol{c}) \; . \tag{1}$$

The above equation also highlights the relation to depth-from-mono architectures [7, 17, 10, 32] which solve a codeless version of the problem, $D = D(I)$. Essentially, the employed architecture is a combination of the depth-from-mono-architecture of Zhou et al. [32] and a variational auto-encoder for depth. We have chosen a variational auto-encoder network [15] in order to increase the smoothness of the mapping between code and depth: small changes in the code should lead to small changes in the depth. While the *practicality* of our representation is thus addressed by the smoothness and the limited code size, the *accuracy* is maximised by training for the reconstruction error.

2.1. Detailed Network Architecture

An overview of the network architecture is provided in Figure 3. The top part illustrates the U-Net [23] applied on the intensity image, which first computes an increasingly coarse but high-dimensional feature representation of the input image. This is followed by an up-sampling part with skip-layers. The computed intensity features are then used to encode and decode the depth in the lower part of the figure. This part is a fairly standard variational auto-encoder architecture with again a down-sampling part and an up-sampling part. Embedded in the middle are two fully connected layers as well as the variational part, which samples the code from a Gaussian distribution and is subject to a regularisation cost (KL-divergence, see [15]). The conditioning of the auto-encoder is achieved by simply concatenating the intensity features of the corresponding resolution.

Instead of predicting just raw depth values, we predict a mean μ and an uncertainty b for every depth pixel. The uncertainty is predicted from intensity only and thus is not directly influenced by the code. Subsequently, we derive a cost term by evaluating the negative log-likelihood of the observed depth $\tilde{d}$. This allows the network to attenuate the cost of difficult regions and to focus on reconstructing parts which can be well explained. At test time, the learned uncertainties can also serve to gauge the reliability of the reconstruction. In the present work we employ a Laplace distribution which has heavier tails than the traditional Gaussian distribution:

$$p(\tilde{d}|\mu, b)) = \frac{1}{2b} \exp\left(-\frac{|\tilde{d} - \mu|}{b}\right) \; . \tag{2}$$

Discarding a constant offset, the negative log-likelihood thus becomes:

$$-\log(p(\tilde{d}|\mu, b)) = \frac{|\tilde{d} - \mu|}{b} + \log(b) \; . \tag{3}$$

Intuitively, the network will tune the pixel-wise uncertainty b such that it best attenuates the reconstruction error $|\tilde{d} - \mu|$ while being subject to a regularisation term $\log(b)$. Using likelihoods as cost terms is a well-established method and has previously been applied to deep learning problems in computer vision [12, 3].

In analogy to previous work, we evaluate the error at multiple resolutions [32]. To this end, we create a depth image pyramid with four levels and derive the negative log-likelihood for every pixel at every level. We increase the weight on every level by a factor of 4 in order to account for

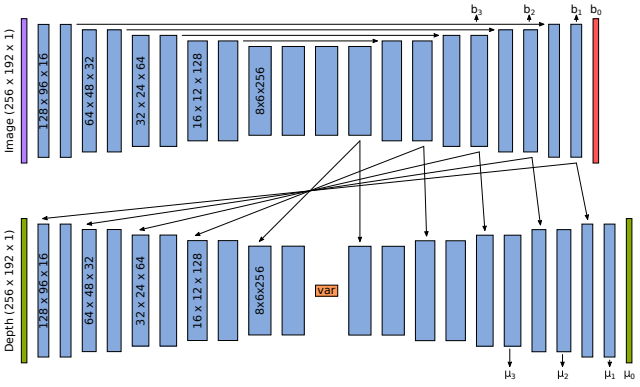

Figure 3. Network architecture of the variational depth auto-encoder conditioned on image intensities. We use a U-Net to decompose the intensity image into convolutional features (the upper part of the figure). These features are then fed into the depth auto-encoder by concatenating them after the corresponding convolutions (denoted by arrows). Down-sampling is achieved by varying stride of the convolutions, while up-sampling uses bilinear interpolation (except for the last layer which uses a deconvolution). A variational component in the bottleneck of the depth auto-encoder is composed of two fully connected layers (512 output channels each) followed by the computation of the mean and variance, from which the latent space is then sampled. The network outputs the predicted mean μ and uncertainty b of the depth on four pyramid levels.

the lower pixel count. Except for the computation of the latent distribution and the output channels, the activations are all set to ReLu. Furthermore, for allowing pre-computation of the Jacobians (see Section 4.1), we explore identity activations for the depth decoder. However, in order to retain an influence from image to code-Jacobian, we add the element-wise multiplication of every concatenation to the concatenation itself. I.e., we increment every concatenation $[L1, L2]$ of layers $L1$ and $L2$ to $[L1, L2, L1 \odot L2]$.

2.2. Training Setup

The depth values of the dataset are transformed to the range $[0, 1]$. We do this by employing a hybrid depth parametrisation which we call *proximity*:

$$p = \frac{a}{d + a} \,. \tag{4}$$

Given an average depth value a, it maps the depth in $[0, a]$ to $[0.5, 1.0]$ (similar to regular depth) and maps the depths in $[a, \infty]$ to $[0, 0.5]$ (similar to inverse depth). This parametrisation is differentiable and better relates to the actual observable quantity (see inverse depth parametrisation [19]).

The network is trained on the SceneNet RGB-D dataset [18] which is composed of photorealistic renderings of randomised indoor scenes. It provides colour and depth images as well as semantic labeling and poses, out of which we only make use of the two former ones. We make use of the ADAM optimiser [14] with an initial learning rate of 10^{-4}. We train the network for 6 epochs while reducing the learning-rate to 10^{-6}.

3. Dense Warping

Due to the latent cost of the variational auto-encoder, the zero code can be used to obtain a likely single view depth prediction $D(I, 0)$ (see Figure 6). However, if overlapping views are available we can leverage stereopsis to refine the depth estimates. This can be done by computing dense correspondences between the views: Given the image I_A and the estimated code $\boldsymbol{c}_A$ of a view A, as well as the relative transformation $\boldsymbol{T}_A^B = (\boldsymbol{R}_A^B, {}_B\boldsymbol{t}_A^B) \in SO(3) \times \mathbb{R}^3$ to a view B, we compute the correspondence for every pixel $\boldsymbol{u}$ with:

$$w(\boldsymbol{u}, \boldsymbol{c}_A, \boldsymbol{T}_A^B) = \pi(\boldsymbol{R}_A^B \, \pi^{-1}(\boldsymbol{u}, D_A[\boldsymbol{u}]) + {}_B\boldsymbol{t}_A^B) \,, \tag{5}$$

where π and π^{-1} are the projection and inverse projection operators. We use the shortcut $D_A = D(I_A, \boldsymbol{c}_A)$ and use square brackets to denote pixel lookup. If applied to intensity images we can for instance derive the following photometric error:

$$I_A[\boldsymbol{u}] - I_B[w(\boldsymbol{u}, \boldsymbol{c}_A, \boldsymbol{T}_A^B)] \,. \tag{6}$$

The above expressions are differentiable w.r.t. to their inputs and we can compute the corresponding Jacobians using the chain rule:

$$\frac{\partial I_B[\boldsymbol{v}]}{\partial {}_B\boldsymbol{t}_A^B} = \frac{\partial I_B[\boldsymbol{v}]}{\partial \boldsymbol{v}} \frac{\partial \pi(\boldsymbol{x})}{\partial \boldsymbol{x}} \,, \tag{7}$$

$$\frac{\partial I_B[\boldsymbol{v}]}{\partial \boldsymbol{R}_A^B} = \frac{\partial I_B[\boldsymbol{v}]}{\partial \boldsymbol{v}} \frac{\partial \pi(\boldsymbol{x})}{\partial \boldsymbol{x}} (-\boldsymbol{R}_A^B \, \pi^{-1}(\boldsymbol{u}, d))^\times \,, \tag{8}$$

$$\frac{\partial I_B[\boldsymbol{v}]}{\partial \boldsymbol{c}_a} = \frac{\partial I_B[\boldsymbol{v}]}{\partial \boldsymbol{v}} \frac{\partial \pi(\boldsymbol{x})}{\partial \boldsymbol{x}} \boldsymbol{R}_A^B \frac{\partial \pi^{-1}(\boldsymbol{u}, d)}{\partial d} \frac{\partial D_A[\boldsymbol{u}]}{\partial \boldsymbol{c}_A} \,, \tag{9}$$

where $^\times$ refers to the skew symmetric matrix of a 3D vector and with the abbreviations:

$$\boldsymbol{v} = w(\boldsymbol{u}, \boldsymbol{c}_A, \boldsymbol{T}_A^B) \,, \tag{10}$$

$$\boldsymbol{x} = \boldsymbol{R}_A^B \, \pi^{-1}(\boldsymbol{u}, D_A[\boldsymbol{u}]) + {}_B\boldsymbol{t}_A^B \,, \tag{11}$$

$$d = D(I_A, \boldsymbol{c}_A)[\boldsymbol{u}]. \tag{12}$$

Most partial derivatives involved in Equations (7) to (9) are relatively well-known from dense tracking literature [13] and include the image gradient ($\partial I_B[\boldsymbol{v}]/\partial \boldsymbol{v}$), the differential of the projection ($\partial \pi(\boldsymbol{x})/\partial \boldsymbol{x}$), as well as transformation related derivatives (also refer to [1] for more details). The last term in Equation (9), $\partial D_A[\boldsymbol{u}]/\partial \boldsymbol{c}_A$, is the derivative of the depth w.r.t. the code. Since it involves many convolutions, it is computationally costly to evaluate (up to 1 sec depending on the size of the network). In case of a linear decoder this term can be pre-computed which significantly accelerates the evaluation of the Jacobians.

4. Inference Framework

4.1. N-Frame Structure from Motion (Mapping)

The proposed depth parametrisation is used to construct a dense N-frame Structure from Motion (SfM) framework (see Figure 4). We do this by assigning an unknown code and an unknown pose to every frame. All codes and poses are initialised to zero and identity, respectively. For two frames A and B with overlapping field of view we then derive photometric and geometric residuals, E_{pho} and E_{geo}, as follows:

$$E_{\text{pho}} = L_p\big(I_A[\boldsymbol{u}] - I_B[w(\boldsymbol{u}, \boldsymbol{c}_A, \boldsymbol{T}_A^B)]\big) \,, \tag{13}$$

$$E_{\text{geo}} = L_g\big(D_A[\boldsymbol{u}] - D_B[w(\boldsymbol{u}, \boldsymbol{c}_A, \boldsymbol{T}_A^B)]\big) \,. \tag{14}$$

The loss functions L_{pho} and L_{geo} have the following masking and weighting functionality: (i) mask invalid correspondences, (ii) apply relative weighting to geometric and photometric errors, (iii) apply a Huber weighting, (iv) downweight errors on strongly slanted surfaces, and (v) downweight pixels which might be occluded (only L_{pho}).

In order to optimise both sets of residuals w.r.t. our motion and geometry we compute the Jacobians w.r.t. all codes and poses according to Section 3. As mentioned above, we investigate the applicability of linear decoding networks

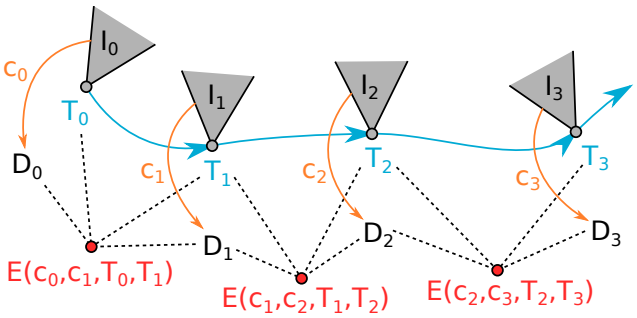

Figure 4. Illustration of the SfM system. The image I_i and corresponding code $\boldsymbol{c}_i$ in each frame are used to estimate the depth D_i. Given estimated poses $\boldsymbol{T}_i$, we derive relative error terms between the frames (photometric and geometric). We then jointly optimise for geometry ($\boldsymbol{c}_i$) and motion ($\boldsymbol{T}_i$) by using a standard second-order method.

(see Section 2.1) as this allows us to compute the Jacobian of the decoder $D(I, \boldsymbol{c})$ w.r.t. the code $\boldsymbol{c}$ only once per keyframe. After computing all residuals and Jacobians we apply a damped Gauss-Newton algorithm in order to find the optimal codes and poses of all frames.

4.2. Tracking (Localisation)

The tracking system, responsible for estimating the pose of keyframes with respect to an existing keyframe map, can be built much in the spirit of the above SfM approach. The current frame is paired with the last keyframe and the estimated relative pose results from a cost-minimisation problem. In our vision-only setup we do not have access to the current depth image (except for a rough guess), and thus in contrast to the described SfM system we do not integrate a geometric cost.

In order to increase tracking robustness we perform a coarse to fine optimisation by first doing the dense alignment on the low depth image resolutions.

4.3. SLAM System

We implement a preliminary system for Simultaneous Localisation and Mapping inspired by PTAM [16] where we alternate between tracking and mapping. The initialisation procedure takes two images and jointly optimises for their relative pose and the codes of each frame. After that we can track the current camera pose w.r.t. the last keyframe. Once a certain baseline is achieved we add a keyframe to the map and perform a global optimisation, before continuing with the tracking. If the maximum number of keyframes is reached we marginalise old keyframes and thereby obtain a linear prior on the remaining keyframes. In a 4-keyframes setup, we achieve a map update rate of 5 Hz, which if we do not have to add keyframes too frequently is enough for real-time performance. The system currently relies on Tensorflow for image warping, and could be sped up with a more targeted warping and optimisation system which are both part of future work.

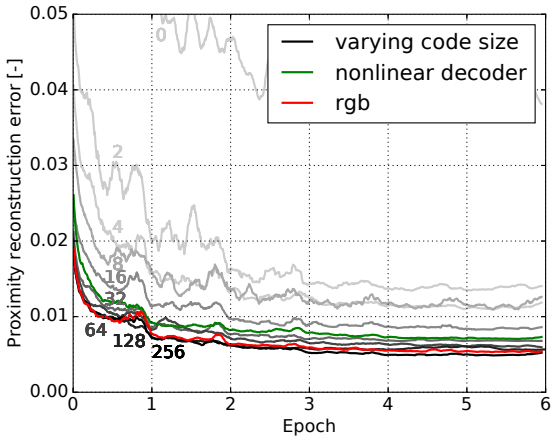

Figure 5. Validation loss during training on the per-pixel proximity errors. As the reference implementation, we use a network trained on greyscale images with a linear decoder. Lower losses can be achieved by increasing the code size (increasing shades of grey). Using a nonlinear decoder or colour images during training does not affect the results in a significant way.

5. Experimental Evaluation and Discussion

Please also see our submitted video which includes demonstrations of our results and system http://www.imperial.ac.uk/dyson-robotics-lab/projects/codeslam/.

5.1. Image Conditioned Depth Encoding

First we present results and insights related to our key concept of encoding depth maps conditioned on intensity images.

We trained and compared multiple variations of our network. Our reference network has a code size of 128, employs greyscale image information only, and makes use of a linear decoder network in order to speed up Jacobian computation. Figure 5 shows results on reconstruction accuracy using different code sizes as well as setups with RGB information and nonlinear depth decoding. The use of colour or nonlinear decoding did not significantly affect the accuracy. With regard to code size, we observe a saturation of the accuracy at a code size of 128; there is little to be gained from making the code bigger. This value may be surprisingly low, but the size seems to be large enough to transmit the information that can be captured in the code by the proposed network architecture.

Figures 6 to 8 provide some insight into how our image

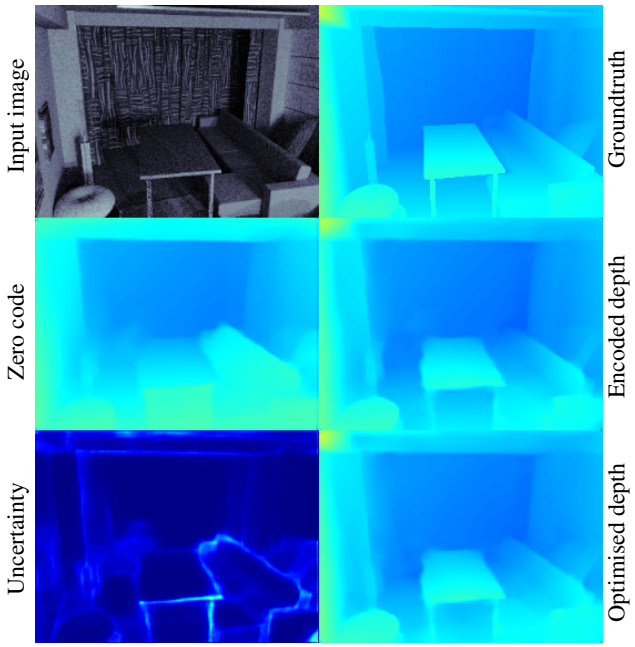

Figure 6. An example image passed through encoding and decoding. Top left: input image. Top right: ground truth depth. Middle left: zero code reconstruction (image only prediction). Middle right: decoded depth (code from encoder). Bottom left: estimated reconstruction uncertainty (scaled four times for visibility). Bottom right: optimised depth (code minimising reconstruction error).

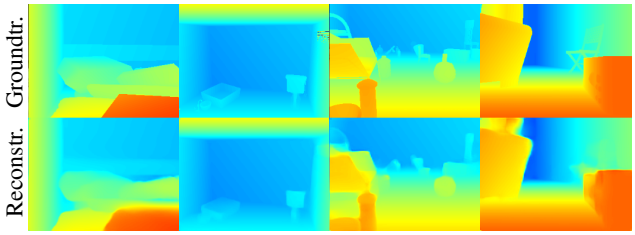

Figure 7. Encodings of different depth images. The encoding allows to capture even fine geometrical details.

conditioned depth encoding works. In Figure 6 we show how we encode a depth image into a code of size 128. Using the corresponding intensity image this can then be decoded into a reconstructed depth image, which captures all of the main scene elements well. We also show the reconstruction when passing a zero code to the decoder as well as with a code that is optimised for minimal reconstruction error. The zero code captures some of the geometrical details but fails to properly reconstruct the entire scene. The reconstruction with the optimised code is very similar to the one with the code from the encoder which indicates that the encoder part of the network works well. The associated depth uncertainty is also visualised and exhibits higher magnitudes

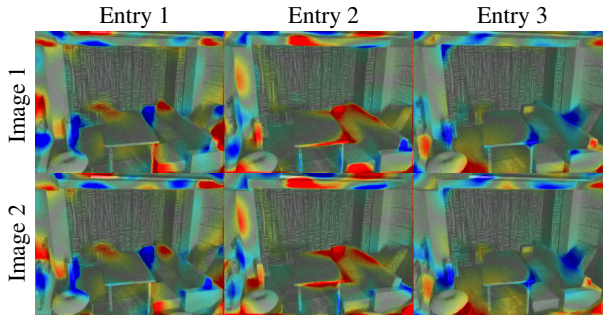

Figure 8. Visualisation of the influence of the code on depth reconstruction. The Jacobian of the depth w.r.t. a specific code entry is used to colourise the input image (blue and red depict negative and positive values, respectively). Columns represent code entries (1-3). Rows represent two different input images.

# frames	1	2	3	4	5	6
RMSE [10^{-2}]	2.65	2.47	2.31	2.39	2.30	2.14

Table 1. RMS of pixel proximity estimation error with different amounts of master keyframe-frame pairs in the optimisation problem. The error is evaluated between the master keyframe proximity and its corresponding ground truth proximity. Frames 1-3: downward-backwards motion. Frames 4-6: left-forward motion.

in the vicinity of depth discontinuities and around shiny image regions (but not necessarily around high image gradients in general). Further examples of depth encoding are shown in Figure 7.

In Figure 8 we visualise the Jacobians of the depth image w.r.t. to the code entries. An interesting observation is that the single code entries seem to correspond to specific image regions and, to some extent, respect boundaries given by the intensity image. While the regions seem to be slightly fragmented, the final reconstructions will always be a linear combination of the effect of all code entries. We also compare the regions of influence for two different but similar images and can observe a certain degree of consistency.

5.2. Structure from Motion

The proposed low dimensional encoding enables continuous refinement of the depth estimates as more overlapping keyframes are integrated. In order to test this, we have implemented an SfM system which incrementally pairs one pre-selected frame with all the remaining frames (which were selected from SceneNet RGB-D). Table 1 shows the obtained reconstruction error w.r.t. the number of frames that are connected to the first frame. The observed reduction of the reconstruction error well illustrates the strength of the employed probabilistic inference method, application of which is enabled by the low dimensionality of the optimisation space. The magnitude of depth refinement depends on the information content of the new frames (whether they

Figure 9. Monocular 3D reconstruction using 9 keyframes. During optimisation a selected master keyframe is paired with the other frames. The depth images of all frames are used for the 3D rendering. The employed geometric error term ensures the consistency between the depth of the different views.

Figure 10. Two-frame SfM on selected pairs from the NYU V2 dataset. Top row presents one of the images used for reconstruction, while the bottom row contains respective depth estimates. The main elements of all scenes can be well perceived in the depth image. The overexposed image regions saturate to infinite depth values, which is a result of using the SceneNet RGB-D dataset for training, which contains many scenes with windows (similar to the one in the left image).

present the scene under a new view and exhibit sufficient baseline). Figure 9 presents a 3D reconstruction based on 9 frames for the scene used in the above error computations. Since in this rendering all the frame depth maps are superimposed, one can observe the quality of the alignment. In a future full SLAM system, these keyframes would be fused together in order to form a single global scene. Before visualisation, high frequency elements are removed from the depth maps with bilateral filtering and highly slanted mesh elements are cropped.

Being exposed to a large variety of depth images during training, the proposed network embeds geometry priors in its weights. These learned priors seem to generalise to real scenes as well: Figure 1 depicts a two-frame reconstruction with images from the real image EuRoC dataset [2] taken by a drone in an industrial setting. The result corresponds to 50 optimisation steps, each taking around 100 ms to complete. Since significant exposure changes occur between the images, we perform an affine illumination correction of the frames. The validation of the two-frame reconstruction performance is of high importance as it is directly connected to the initialisation procedure of the full SLAM system. In order to further highlight its effectiveness we include results on a selection of pairs taken from the NYU V2 dataset [27] (Figure 10).

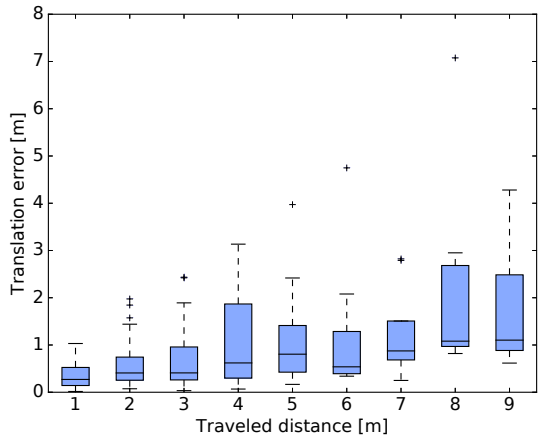

Figure 11. Translation error versus traveled distance on the EuRoC dataset MH02. Despite training the auto-encoder on SceneNet RGB-D, its decoder generalises to other datasets (after correcting for camera intrinsics).

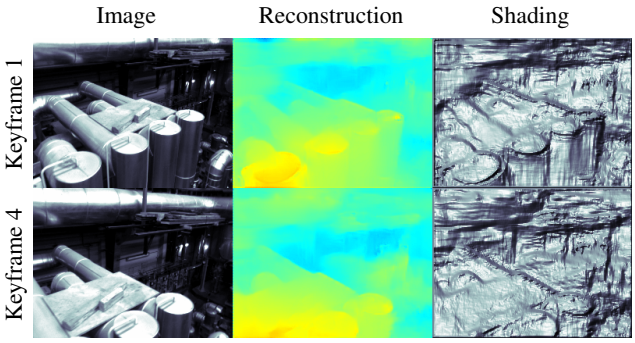

Figure 12. Example structure from motion results on frames from the EuRoC dataset. From the left: image, estimated proximity, shaded proximity.

5.3. SLAM System

In contrast to most dense approaches, our low dimensional geometry encoding allows joint optimisation of motion and geometry. Furthermore, due to the inherent prior contained in the encoding, the framework is able to deal with rotational motions only. The system is tested in a sliding window visual odometry mode on the EuRoC dataset on trajectory MH_02_easy. Even though the dataset is significantly different from the data the network is trained on (with many metallic parts and many reflections), the proposed system is able to run through most of this arguably very difficult dataset (we do not use the available IMU data).

Figure 11 shows the error against traveled distance. While this cannot compete with a state-of-the art visual-inertial system, it performs respectably for a vision only-system and exhibits an error of roughly 1 m for a traveled distance of 9 m. In Figure 12 the first and last key-frame of our 4-frame sliding window system are illustrated. This shows the intensity image of the encountered scene together with the estimated proximity image and a normal based shading. Considering that the network was trained on artificial images only which were very different in their nature, the reconstructed depth is sensible and allows for reliable camera tracking.

6. Conclusions

We have shown that a learned representation for depth which is conditioned on image data provides an important advance towards future SLAM systems. By employing an auto-encoder like training setup, the proposed representation can contain generic and detailed dense scene information while allowing efficient probabilistic joint optimisation together with camera poses.

In near future work, we will use the components demonstrated here to build a full real-time keyframe-based SLAM system. Learned visual motion estimation methods could surely be brought in here as priors for robust tracking. In addition to that, the training of the network should be extended in order to include real data as well. This could be done by using an RGB-D dataset, but might also be achieved with intensity information only in an self-supervised manner, based on photometric error as loss.

In the longer term, we would like to move beyond a keyframe-based approach, where our dense geometry representations are tied to single images, and work on learned but optimisable compact representations for general 3D geometry, eventually tying our work up with 3D object recognition.

7. Acknowledgements

Research presented in this paper has been supported by Dyson Technology Ltd.

References

[1] M. Bloesch, H. Sommer, T. Laidlow, M. Burri, G. Nützi, P. Fankhauser, D. Bellicoso, C. Gehring, S. Leutenegger, M. Hutter, and R. Siegwart. A Primer on the Differential Calculus of 3D Orientations. *CoRR*, abs/1606.0, 2016. 4

[2] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC Micro Aerial Vehicle Datasets. *International Journal of Robotics Research (IJRR)*, 35(10):1157–1163, September 2016. 7

[3] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. VidLoc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[4] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. VINet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2017. 2

[5] A. J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2003. 1

[6] H. Durrant-Whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006. 1

[7] D. Eigen, C. Puhrsch, and R. Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *Neural Information Processing Systems (NIPS)*, 2014. 3

[8] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017. 1

[9] J. Engel, T. Schoeps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 1, 2

[10] R. Garg, V. K. B. G, G. Carneiro, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2, 3

[11] M. Kaess. Simultaneous localization and mapping with infinite planes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015. 2

[12] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Neural Information Processing Systems (NIPS)*, 2017. 3

[13] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013. 4

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 4

[15] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 3

[16] G. Klein and D. W. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007. 1, 2, 5

[17] F. Liu, C. Shen, and G. Lin. Deep Convolutional Neural Fields for Depth Estimation from a Single Image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3

[18] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison. SceneNet RGB-D: Can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation? In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017. 4

[19] J. M. M. Montiel, J. Civera, and A. J. Davison. Unified Inverse Depth Parametrization for Monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2006. 4

[20] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics (T-RO)*, 31(5):1147–1163, 2015. 1

[21] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 1

[22] L. Platinsky, A. J. Davison, and S. Leutenegger. Monocular visual odometry: Sparse joint optimisation or dense alternation? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 1

[23] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2015. 3

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. 2

[25] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison. Dense planar SLAM. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2014. 2

[26] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[27] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. 7

[28] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[29] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. DeMoN: Depth and motion network for learning monocular stereo. *arXiv preprint arXiv:1612:02401*, 2016. 2

[30] S. Wang, R. Clark, H. Wen, and N. Trigoni. DeepVO: Towards end to end visual odometry with deep recurrent convolutional neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 2

[31] Z. Yin and J. Shi. GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[32] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3