Unsupervised Learning of Depth and Visual Odometry using Photometric Calibration

Tianhong Yu
Beijing Institute of Technology
tedyth@163.com

Abstract. Depth and visual odometry estimation are two essential parts in SLAM systems. Compared with traditional algorithms, supervised learning methods have shown promising results in single view depth estimation and visual odometry estimation. However, they require large amounts of labeled data. Recently, some unsupervised approaches to estimate depth and odometry via minimizing photometric error draw great attention. In this paper, we present a novel approach to learn depth and odometry via unsupervised learning. Our method ameliorates the original photometric loss to enhance the robustness to illumination change in real scenarios. In addition, we propose a new structure of Pose-net and Explainability-net to achieve rotation-sensitive odometry results and more accurate explainability masks. The experimental results have demonstrated that our approach achieves better performance than existing unsupervised methods in both depth and odometry results.

1. Introduction
In recent years, SLAM (Simultaneous Localization And Mapping) algorithms have been widely used in automatic-driving cars or other autonomous robots. Visual odometry, as a part of SLAM, learns camera poses with only video frames as input. On the other hand, depth prediction is also an essential part in SLAM system. An accurate depth map is needed when creating a large 3D point cloud.

Since these two parts are indispensable, several works [1][2][3][4] estimate depth using the supervised method. However, it requires large amounts of labeled data, which is difficult to obtain.

Unsupervised methods, proposed by [9], build a depth network and a pose network. They utilize photometric warp error as the training loss to jointly train the above two networks. On the basis of [9], a number of works [10] [11] [12] [13] [14] are proposed and achieve better results. However, there is still a problem in [9], that is, the photometric loss is easily affected by light changes.

Inspired by that, in this paper, we propose a framework (figure 1) which jointly estimates depth and camera poses based on [9]. We make the following contributions. 1)We apply photometric calibration to each frame. Specifically, we propose a novel loss based on the brightness of each video frames, which solves the problem of the illumination change in the real environment. 2)Two groups of separate convolutional layers are proposed in Pose-net in order to balance the weights between translation and rotation loss. 3)We apply a convolutional layer and a deconvolutional layer in our explainability network, which selectively enhances useful features and suppresses less useful ones. 4)The experimental results demonstrate that we significantly improve the performance over the existed methods in both depth estimation and odometry estimation.
2. Our Approach
In this section, we introduce our network which learns depth and camera poses from consecutive video frames in an unsupervised method.

2.1. Network structure
We mainly adopt the architecture of [9]. Our work consists of three networks, Depth-net, Pose-net and Explainability-net.

Depth-net is an encoder-decoder network similar to [15]. It takes a single image as input and output four scales of dense depth maps. The structure of Pose-net and Explainability-net is in figure 2. As we can see in figure 2, Pose-net is a simple convolution network like VGGNet in [16]. We input consecutive camera frames (3 frames or 5 frames) and output 6-DoF camera poses of each frame.

At the same time, we take full advantage of the first few layers in Pose-net and add some convolution and deconvolution layers behind them to get our Explainability-net. By means of the deconvolution layers, we obtain explainable masks in four scales. These masks will be used in computing the training loss, which will be introduced in the subsequent chapter.

2.2. A Brightness-based loss
In [9], we assume that the surface is Lambertian so that the photo-consistency error is meaningful. However, the brightness of the scene varies while the camera moves in reality. This will cause a problem that the photometric error is not accurate. Inspired by the idea of making a photometric calibration in [11], we propose a photometric error based on the brightness of the scene which is robust to the change of brightness.

Our problem can be formalized as follows: Our network input is consecutive image sequence (take 3 as an example) \( I_{t-1}, I_t, I_{t+1} \), including the source view and target view. Our goal is to obtain the depth \( D_t \) of each frame and camera poses \( T_{t-1}, T_t, T_{t+1} \), representing the camera ego-motion. Then, according to [9], the pixel loss can be formulated as follows:

\[
L_{\text{pixel}} = \sum_s \sum_p \left| I_s(p) - I_t(p) \right|
\]  

where \( p \) represents pixel coordinates in the \( I \) image. \( I_t(p) \) is the target view image. \( I_s(p) \) is the warped image from the target view to the source view. The formula about warping is as follows:

\[
p_s = KT_{t \rightarrow s}^{-1} D_{t}(p_t) K^{-1} p_t
\]
where \( p_t \) represents pixel coordinates in the target image. \( D_t(p_t) \) denotes the depth of \( p_t \) which is obtained by the Depth-net. \( K \) denotes the intrinsic matrix of the camera. \( T \) is the transformation from the target view to the source view obtained by the Pose-net.

However, this loss function suffers when there are significant illumination changes in the video. To solve this problem, we propose a new pixel loss instead of the original one. We use \( B(p) \) to denote the brightness of one pixel in one video frame \( I \). \( B(p) \) can be represented as:

\[
B(p) = 0.6 \times r(p) + 0.3 \times g(p) + 0.1 \times b(p)
\]  

where \( r(p), g(p), b(p) \) respectively represents the pixel value in red, green, blue channels of the image \( I \).

Then, the pixel loss can be formulated as follows:

\[
L_{\text{pixel}} = \sum_{i} \sum_{p} |X_i(p) - X_s(p)|
\]  

(3)

We use \( X(p) \) to describe pixel \( p \) in the image \( I \) without the influence of brightness. Therefore, we can use the difference between \( X(p) \) to replace the original pixel loss.

According to the method in [9]'s work, in order to improve the performance against occlusion and moving objects problem, we add a predicted mask obtained by the network in figure 2 on the pixel loss, our final loss consists of pixel loss, smooth loss and explainability loss. It can be formulated as follows:

\[
L_{\text{final}} = \sum_{l} L_{\text{pixel}}^l + \lambda_s L_{\text{smooth}}^l + \lambda_e \sum_s L_{\text{reg}}^l (E_s^l)
\]  

(4)

where \( l \) represents different image scales, \( s \) represents source images, and \( \lambda_s \) and \( \lambda_e \) are respectively the weights of smooth loss and the explainability loss. You can find the introduction of smooth loss and explainability loss in [9]'s work.

We use this final loss to jointly train the Depth-net and Pose-net and achieve a better performance we’ll see in the next part.

2.3. Rotation-sensitive Pose-net
In most cases, the camera follows the car's movement which consists of translation and rotation. However, translation is more common than rotation in reality. As a result, this will cause a problem that translation is easy to fit during training time while the rotation has not been trained completely yet. In other words, the original network is not sensitive to rotation.

The above analysis motivates us to investigate the problem of the asynchrony of translation and rotation. We change the structure of the network in [9]'s work. We regard rotation and translation as two irrelevant results respectively. Specifically, we apply two groups of two separate convolutional layers before the last camera pose vectors, which can be seen in figure 2. In this way, there are no relationships between our Pose-net's results (rotation and translation), since they are trained separately. Thus we can get a more accurate rotation result throughout our rotation-sensitive network. Because of the more accurate rotation result, depth results are improved subsequently, which we'll find out in Section 3.

2.4. Improved Explainability-net
It is widely known that getting good performance via a deep convolutional network is a complicated task. In recent works, one way to solve this problem is to improve the ability of distinguishing the features.
In our work, the Explainability-net shares the first five feature encoding layers with Pose-net. After these five convolutional layers, we add a convolutional layer to get distinctive features. It is then followed by four deconvolutional layers, which results in four scales of explainable masks. We simply apply a convolutional layer after the feature map and append a deconvolutional layer subsequently to recover the scale of the feature map. In other words, we suppress our network model in the spatial dimension at first to distinguish the more important channel and enhance the ability of feature expression. Then we recover to the original spatial dimension so that the feature maps carry more important information.

3. Experimental Results

In this section, we will show experimental results for evaluating the performance of our proposed framework. Our network is implemented with the Tensorflow framework. We use GTX 1080Ti to train and test our network. We achieve real-time performance during test time.

3.1. Training configurations

**Dataset:** We follow [9]'s work and choose KITTI dataset to train and test our network.

**Training parameters:** We choose the Adam optimizer [17] as the optimizer of our framework. We utilize a small learning rate of 0.0001, \( \beta_1 = 0.9 \) and \( \beta_2 = 0.999 \). we set the batch size for training to 4 in order to improve the training speed.

3.2. Depth Estimation

Here, we compare our depth estimation results with other various methods to evaluate the performance of our pose estimation network. We fix the length of image sequences to be 3 frames, and treat the central frame as the target view and the \( \pm 1 \) frames as the source views. We choose the split provided by [2], which has been mentioned before.

The detailed depth estimation results are listed in table 1. Some comparison results are even supervised method. Part of results are not only trained in one dataset KITTI, but also trained on two datasets, KITTI and Cityscapes [18]. All the methods are tested in KITTI dataset. The table reports separate results for a depth cap of 50m and 80m. As we can see from the table, our method outperforms most of the methods mentioned in the table. Our results are even better than some supervised methods or methods trained on both KITTI and Cityscapes [18] dataset.

| Method            | Dataset | Supervision | Cap |   |    | Error Metric | Accuracy Metric |
|-------------------|---------|-------------|-----|---|----|---------------|----------------|
|                   |         |             |     |   |    | Abs-Rel | Sq-Rel | RMSE | Log-RMSE | \( \delta<1.25 \) | \( \delta<1.25^2 \) | \( \delta<1.25^3 \) |
| Eigen et al.[2]   | K       | Depth       | 80m | 0.214 | 1.605 | 6.563 | 0.292 | 0.673 | 0.884 | 0.957 |
| Coarse            |         |             |     |     |     |     |     |     |     |     |
| Eigen et al.[2]   | K       | Depth       | 80m | 0.203 | 1.548 | 6.307 | 0.282 | 0.702 | 0.890 | 0.958 |
| Fine              |         |             |     |     |     |     |     |     |     |     |
| Liu et al.[4]     | K       | Depth       | 80m | 0.202 | 1.614 | 6.523 | 0.275 | 0.678 | 0.895 | 0.965 |
| Godard et al.[6]  | K       | Stereo      | 80m | 0.148 | 1.344 | 5.927 | 0.247 | 0.803 | 0.922 | 0.964 |
| Stereop           |         |             |     |     |     |     |     |     |     |     |
| Godard et al.[6]  | CS+K    | Stereo      | 80m | 0.124 | 1.076 | 5.311 | 0.219 | 0.847 | 0.942 | 0.973 |
| Zhou et al.[9]    | K       | -           | 80m | 0.208 | 1.768 | 6.856 | 0.283 | 0.678 | 0.885 | 0.957 |
3.3. Odometry Estimation

As more source views express more temporal information and perform better in odometry estimation experiments, we fix the length of image sequences to be 5 frames, and treat the central frame as the target view and the ±2 frames as the source views. We choose the official KITTI Odometry set as the training set, which has been mentioned before.

We show our odometry estimation results in table 2. ORB-SLAM is a classical monocular SLAM system proposed by [19]. ORB-SLAM(full) is the complete version of ORB-SLAM, including loop closure and relocalization. ORB-SLAM(short) denotes the system runs on 5-frame snippets (same as our input setting). For the sake of scale consistency, we optimize the scaling factor in advance. From the data in the table, we can discover that our method significantly outperforms our baseline [9]. And our method’s performance is very close to the performance of ORB-SLAM and [14]’s method.

Table 2. Odometry estimation results on the KITTI Odometry set. We adopt ATE (Absolute Trajectory Error) proposed in [19] as our evaluation metric. Lower is better. Our method significantly outperforms our baseline [9] and is very close to the performance of ORB-SLAM and [14]’s method.

| Method             | Sequence 9  | Sequence 10 |
|--------------------|-------------|-------------|
| ORB-SLAM[19](full) | 0.014±0.008 | 0.012±0.011 |
| ORB-SLAM[19](short)| 0.064±0.141 | 0.064±0.130 |
| Zhou et al.[9]     | 0.021±0.017 | 0.020±0.015 |
| Mahjourian et al.[14]| 0.013±0.010 | 0.012±0.011 |
| **Ours**           | **0.014±0.008** | **0.013±0.010** |

4. Conclusion

We present a novel framework to learn depth and odometry using the unsupervised method. Our contribution is to ameliorate the original photometric loss to enhance the robustness to illumination change and propose a new structure to achieve better results. The proposed method only needs monocular video streams for training, and can provide depth and odometry results during test time.

The results of depth estimation in experiments demonstrate that our algorithm can even outperform some supervised methods and original unsupervised method. Our results of odometry estimation can keep up with traditional SLAM algorithm in some area and can be employed in real time.
Finally, it is possible to employ our framework into a real SLAM system with loop detection. This can significantly reduce drift and improve performance.

References

[1] P. Agrawal, J. Carreira, and J. Malik, “Learning to see by moving,” in IEEE International Conference on Computer Vision, 2015, pp. 37–45

[2] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in International Conference on Neural Information Processing Systems, 2014, pp. 2366–2374

[3] F. Liu, C. Shen, and G. Lin, “Deep convolutional neural fields for depth estimation from a single image,” pp. 5162–5170, 2014

[4] F. Liu, C. Shen, G. Lin, and I. Reid, “Learning depth from single monocular images using deep convolutional neural fields,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 10, pp. 2024–2039, 2016

[5] R. Garg, K. B. G. Vijay, G. Carneiro, and I. Reid, “Unsupervised cnn for single view depth estimation: Geometry to the rescue,” pp. 740–756, 2016

[6] C. Godard, O. M. Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in Computer Vision and Pattern Recognition, 2017, pp. 6602–6611

[7] Y. Kuznietsov, J. Steckler, and B. Leibe, “Semi-supervised deep learning for monocular depth map prediction,” pp. 2215–2223, 2017

[8] M. Ye, E. Johns, A. Handa, L. Zhang, P. Pratt, and G. Z. Yang, “Selfsupervised siamese learning on stereo image pairs for depth estimation in robotic surgery,” 2017

[9] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” pp. 6612–6619, 2017

[10] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, “Learning depth from monocular videos using direct methods,” 2018

[11] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PP, no. 99, pp. 1–8, 2016

[12] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, “Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction,” 2018

[13] R. Li, S. Wang, Z. Long, and D. Gu, “Undeepvo: Monocular visual odometry through unsupervised deep learning,” 2017

[14] R. Mahjourian, M. Wicke, and A. Angelova, “Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints,” 2018

[15] N. Mayer, E. Ilg, P. Husser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” in Computer Vision and Pattern Recognition, 2016, pp. 4040–4048

[16] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in ICLR, 2015, Conference Proceedings

[17] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” Computer Science, 2014

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” 2016

[19] R. Mur-Artal and J. D. Tardos, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” IEEE Transactions on Robotics, vol. PP, no. 99, pp. 1–8, 2016