CAMERA CALIBRATION THROUGH CAMERA PROJECTION LOSS

Talha Hanif Butt, Murtaza Taj

Computer Vision and Graphics Lab
Lahore University of Management Sciences, Lahore, Pakistan
thanifbutt@gmail.com, murtaza.taj@lums.edu.pk
https://cvlab.lums.edu.pk/

ABSTRACT

Camera calibration is a necessity in various tasks including 3D reconstruction, hand-eye coordination for a robotic interaction, autonomous driving, etc. In this work we propose a novel method to predict extrinsic (baseline, pitch, and translation), intrinsic (focal length and principal point offset) parameters using an image pair. Unlike existing methods, instead of designing an end-to-end solution, we proposed a new representation that incorporates camera model equations as a neural network in a multi-task learning framework. We estimate the desired parameters via novel *camera projection loss* (CPL) that uses the camera model neural network to reconstruct the 3D points and uses the reconstruction loss to estimate the camera parameters. To the best of our knowledge, ours is the first method to jointly estimate both the intrinsic and extrinsic parameters via a multi-task learning methodology that combines analytical equations in learning framework for the estimation of camera parameters. We also proposed a novel CVGL Camera Calibration dataset using CARLA Simulator [1]. Empirically, we demonstrate that our proposed approach achieves better performance with respect to both deep learning-based and traditional methods on 8 out of 10 parameters evaluated using both synthetic and real data. Our code and generated dataset are available at https://github.com/thanif/Camera-Calibration-through-Camera-Projection-Loss.

Index Terms— Camera Projection Loss, Multi-task learning, Camera Calibration, Camera Parameters

1. INTRODUCTION

Camera calibration deals with finding the five intrinsic (focal length, image sensor format, and principal point) and six extrinsic (rotation, translation) parameters of the specific camera.

Most of the existing methods usually ignore the underlying mathematical formulation of the camera model and instead propose an end-to-end framework to directly estimate the desired parameters [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Thus they are difficult to interpret for real-world applications and have so far been able to mainly estimate the focal length of the camera via single image only [8, 9, 10]. The major contributions of our work are as follows:

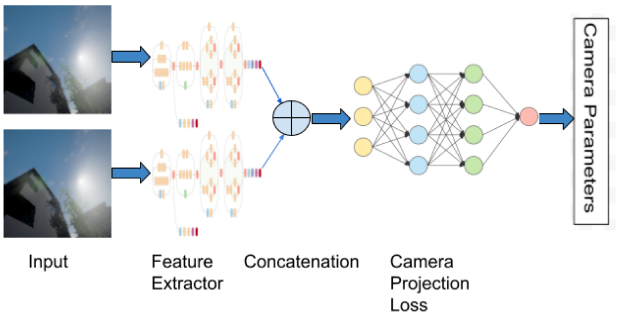


Fig. 1. Our method estimates extrinsic (baseline, pitch and translation) and intrinsic (focal length and principal point offset) parameters using pre-trained Inception-v3 [2] and the proposed Camera Projection Loss.

- To the best of our knowledge, this work is the first learning-based method to jointly estimate both intrinsic and extrinsic camera parameters including camera baseline, disparity, pitch, translation, focal length and principal point offset.
- The existing learning based approaches [8, 3, 10] have not been applied to the estimation of all 10 camera parameters due to lack of any dataset. We addressed this limitation by generating a synthetic dataset from two towns in CARLA [1] simulation consisting of 49 different camera settings.
- Unlike existing methods, instead of designing an end-to-end solution to directly estimate the desired parameters [8], we proposed a new representation that represents camera model equations as a neural network in a multi-task learning (MTL) framework.
- We proposed a novel *camera projection loss* (CPL) that combines analytical equations in learning framework.

2. PROPOSED METHOD

We propose to utilize multi-task learning by incorporating mathematical equations through a new loss function embedded as a neural network for better representation while learning. We train a convolutional neural network to predict the extrinsic and intrinsic camera parameters. To achieve this, we use dependent regressors that share a common network architecture as the feature extractor. We use a Inception-v3 [2] pretrained on ImageNet [13] as a feature extractor followed by the Lambda layers for loss computation with 13 regressors, 10 of which correspond to the camera parameters while 3 correspond to the 3D point cloud. Instead of training these regressors to predict the focal length, principal point, baseline, pitch, and translation, we use proxy variables that are not visible in the image and are dependent on each other. This allows us to directly relate our method with the mathematical foundations of multi-view geometry [14] resulting in better performance.

Camera Model: The camera model that we consider is the perspective projection model based on the pinhole camera [15]. In this work, we rely on projection of 2D image points (u, v) to 3D world points X, Y, Z as a reference. This leaves us with with the estimation of 13 free parameters: focal length (f_x, f_y), principal point (u_0, v_0), disparity (d), baseline (b), pitch (θ_p), translation $\mathbf{t} = \{t_x, t_y t_z\}$ and 3D coordinates (X, Y, Z).

Parameterization: As revealed by previous work [3, 11, 5], an adequate parameterization of the variables to predict can benefit convergence and final performance of the network. For the case of camera calibration, parameters such as the focal length or the tilt angles are difficult to interpret from the image content. Instead, they can be better represented by proxy parameters that are directly observable in the image. We use 2D to 3D projection as a proxy for our parameters.

A 2D point in image coordinate system is projected to camera coordinate and then to world coordinate system and the process can be explained by the following:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \sim \left[\begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T_{3\times1} & 1 \end{pmatrix} \right]^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \quad (1)$$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \sim \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T_{3\times1} & 1 \end{pmatrix}^{-1} \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}. \quad (2)$$

where $\mathbf{R}$ is the camera rotation matrix. The image point (u, v) to camera point $(x_{cam}, y_{cam}, z_{cam})$ transformation can be performed as follows (assuming skew = 0):

$$\begin{pmatrix} y_{cam} \\ z_{cam} \\ x_{cam} \end{pmatrix} \sim \begin{pmatrix} \frac{1}{f_x} & 0 & \frac{-u_0}{f_x} \\ 0 & \frac{1}{f_y} & \frac{-v_0}{f_y} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad (3)$$

$$y_{cam} = \frac{u}{f_x} - \frac{u_0}{f_x} = \frac{u - u_0}{f_x} \quad (4a)$$

$$z_{cam} = \frac{v}{f_y} - \frac{v_0}{f_y} = \frac{v - v_0}{f_y} \quad (4b)$$

$$x_{cam} = 1 \quad (4c)$$

Similarly, for camera to world transformation we have:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \sim \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T_{3\times1} & 1 \end{pmatrix} \begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ 1 \end{pmatrix} \quad (5)$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \sim \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{pmatrix} + \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (6)$$

$$X = x_{cam} * \cos\theta + z_{cam} * \sin\theta + x \quad (7a)$$

$$Y = y_{cam} + y \quad (7b)$$

$$Z = -x_{cam} * \sin\theta + z_{cam} * \cos\theta + z \quad (7c)$$

To project a point from image to camera coordinate:

$$x_{cam} = f_x * b/d \quad (8a)$$

$$y_{cam} = -(x_{cam}/f_x) * (u - u_0) \quad (8b)$$

$$z_{cam} = (x_{cam}/f_y) * (v_0 - v) \quad (8c)$$

x_{cam} works as a proxy for f_x, baseline and disparity while y_{cam} works as a proxy for f_x, u and u_0 and z_{cam} works as a proxy for f_y, v and v_0. Using x_{cam}, y_{cam} and z_{cam} from Eq. 8a, Eq. 8b and Eq. 8c respectively, points can be projected to world coordinate system using:

$$X = x_{cam} * \cos(\theta_p) + z_{cam} * \sin(\theta_p) + t_x \quad (9a)$$

$$Y = y_{cam} + t_y \quad (9b)$$

$$Z = -x_{cam} * \sin(\theta_p) + z_{cam} * \cos(\theta_p) + t_z \quad (9c)$$

X works as a proxy for pitch and t_x while Y works as a proxy for t_y and Z works as a proxy for pitch and t_z.

Camera Projection Loss: When a single architecture is trained to predict parameters with different magnitudes, special care must be taken to weigh the loss components such that the estimation of certain parameters do not dominate the

learning process. We notice that for the case of camera calibration, instead of optimizing the camera parameters separately, a single metric based on 2D to 3D projection of points can be used.

Given two images with known parameters $\omega = (f_x, f_y, u_0, v_0, b, d, \theta_p, t_x, t_y, t_z, X, Y, Z)$ and a prediction of such parameters given by the network $\hat{\omega}$=($f_x^{'}$, $f_y^{'}$, $u_0^{'}$, $v_0^{'}$, $b^{'}$, $d^{'}$, $\theta_p^{'}$, $t_x^{'}$, $t_y^{'}$, $t_z^{'}$, $X^{'}$, $Y^{'}$, $Z^{'}$), we get the projected point in world coordinate system through Eq. 8a - Eq. 9c. Loss is computed between actual ω and predicted $\hat{\omega}$ using:

$$L(\omega, \hat{\omega}) = (\frac{1}{n}) \sum_{i=1}^{n} MAE(\omega, \hat{\omega}) \tag{10}$$

Separating sources of loss errors: The proposed loss solves the task balancing problem by expressing different errors in terms of a single measure. However, using several camera parameters to predict the 3D points introduces a new problem during learning: the deviation of a point from its ideal projection can be attributed to more than one parameter. In other words, an error from one parameter can backpropagate through the camera projection loss to other parameters.

To avoid this problem, we disentangle the camera projection loss, evaluating it individually for each parameter similar to [6]:

$$\begin{aligned}
L_{f_x} &= L((f_x, f_y^{GT}, u_0^{GT}, v_0^{GT}, b^{GT}, d^{GT}, \theta_p^{GT}, \\
&\quad t_x^{GT}, t_y^{GT}, t_z^{GT}, X^{GT}, Y^{GT}, Z^{GT}), \omega) \\
L_{f_y} &= L((f_x^{GT}, f_y, u_0^{GT}, v_0^{GT}, b^{GT}, d^{GT}, \theta_p^{GT}, \\
&\quad t_x^{GT}, t_y^{GT}, t_z^{GT}, X^{GT}, Y^{GT}, Z^{GT}), \omega) \\
&\dots \\
L_Z &= L((f_x^{GT}, f_y^{GT}, u_0^{GT}, v_0^{GT}, b^{GT}, d^{GT}, \theta_p^{GT}, \\
&\quad t_x^{GT}, t_y^{GT}, t_z^{GT}, X^{GT}, Y^{GT}, Z), \omega)
\end{aligned}$$

$$L^* = \frac{L_{f_x} + L_{f_y} + L_{u_0} + ... + L_Z}{13} \tag{11}$$

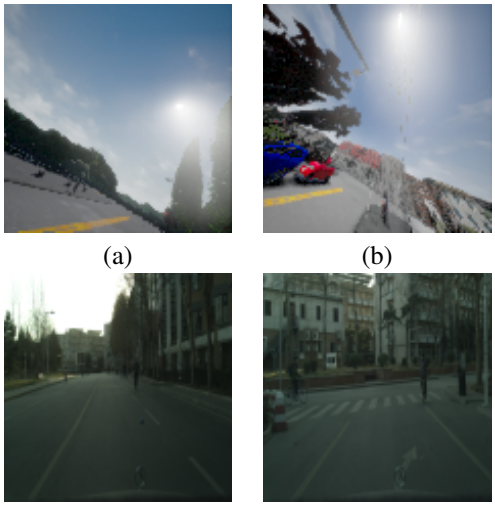
The loss function is further normalized to avoid the unnecessary bias due to one or more error terms by introducing weights α_i with each of the parameters. This bias is introduced due to heterogeneous ranges of various parameters. These weights α_i are learned adaptively during the training process. The updated loss function is defined as:

$$L^* = \alpha_{f_x} L_{f_x} + \alpha_{f_y} L_{f_y} + \alpha_{u_0} L_{u_0} + ... + \alpha_Z L_Z \tag{12}$$

3. RESULTS AND EVALUATION

3.1. Datasets

Synthetic Data: We trained and evaluated our proposed approach by generating a new CVGL Camera Calibration dataset using Town 1 and Town 2 of CARLA [1] Simulator. The dataset consists of 49 camera configurations with town 1 having 25 configurations while town 2 having 24 configurations . The parameters modified for generating the configurations include fov, x, y, z, pitch, yaw, and roll. Here, fov is the field of view, (x, y, z) is the translation while (pitch, yaw, and roll) is the rotation between the cameras. The total number of image pairs is $79,320$, out of which $18,083$ belong to Town 1 while $61,237$ belong to Town 2, the difference in the number of images is due to the length of the tracks.



(a) (b) (c) (d)

Fig. 2. Some representative images from the synthetic and real datasets. (a) Town 1 - CARLA (b) Town 2 - CARLA (c-d) Tsinghua-Daimler.

Real Data: We have used a recent Cyclist Detection dataset [16] for evaluating our approach on real world data. We have used the test set provided by the authors containing $2,914$ images by first deriving the right image using left and disparity images and then use the pair as input to compare different methods.

Implementation Details: Our loss is implemented and trained using Keras [17], an open-source deep learning framework. All networks are trained on GeForce GTX 1050 Ti GPU for 200 epochs with early stopping using ADAM optimizer [18] with Mean Absolute Error (MAE) loss function and a base learning rate η of 10^{-3} with a batch size of 16.

The Lambda layer exists in Keras so that arbitrary expressions can be used as a Layer when constructing Sequential and Functional API models. In the proposed architecture, Lambda layers have been utilized for basic operations including addition, subtraction, multiplication, division, negation, cosine and sine. The intuition is to incorporate mathematical equations in the learning framework.

We have used Normalized Mean Absolute Error for evaluation as follows:

Table 1. Table showing Normalized MAE in predicted parameters on CVGL test set comprising of 23,796 images.

	f_x	f_y	u_0	v_0	b	d	t_x	t_y	t_z	θ_p
Average [3]	0.840	0.786	0.432	0.542	6.552	3.607	6.552	9.372	5.361	0.744
Deep-Homo [8]	0.062	0.062	0.008	0.008	0.156	0.065	0.156	0.161	0.155	0.045
MTL-CPL-U (Ours)	0.935	0.685	0.892	0.737	0.938	0.423	0.400	0.329	0.432	1.060
MTL-Baseline (Ours)	0.030	0.029	0.017	0.007	**0.057**	0.013	**0.064**	**0.076**	**0.071**	0.024
MTL-CPL-A (Ours)	**0.022**	**0.022**	**0.004**	**0.006**	0.093	**0.007**	0.097	0.116	0.098	**0.017**

Table 2. Table showing Normalized MAE in predicted parameters on Tsinghua-Daimler test set comprising of 2,914 images. For this experiment, we just did a forward pass without any transfer learning or training.

	f_x	f_y	u_0	v_0	b	d	t_x	t_y	t_z	θ_p
Average [3]	0.994	0.991	0.969	0.951	112.438	**0.492**	10.843	271.935	13.798	**982.413**
Deep-Homo [8]	0.958	0.958	0.946	0.895	9.985	1.233	0.166	27.141	0.862	2746.994
MTL-CPL-U (Ours)	**0.872**	**0.888**	**0.782**	**0.795**	**0.081**	1.271	**0.147**	**23.836**	**0.635**	7700.968
MTL-Baseline (Ours)	0.957	0.958	0.944	0.893	18.323	1.258	1.035	32.946	0.999	2418.250
MTL-CPL-A (Ours)	0.938	0.938	0.946	0.895	14.182	1.259	0.727	30.640	1.418	1995.353

$$NMAE(y, \hat{y}) = \frac{MAE(y, \hat{y})}{\frac{1}{n}\sum_{i=1}^{n}|y_i|} = \frac{MAE(y, \hat{y})}{mean(|y|)} \tag{13}$$

where y and $\hat{y}$ are the target and estimated values respectively.

3.2. Comparative Analysis

Experimental Setup: We compared our proposed method with two state-of-the-art approaches namely Average field of view [3] and Deep-Homo [8]. Average field of view [3] is a baseline approach, given a query image, it uses the average field of view of the training set as the prediction [3]. Deep-Homo [8] estimates an 8-degree-of-freedom homography between two images. We have modified Deep-Homo[8] to predict the required 13 parameters for comparison purposes as by default, it only predicted 8 values corresponding to the four corners and then using 4-point parameterization and then convert it into the homography matrix. For the purpose of the ablative study, we also created three variants of our multi-task learning approach namely MTL-Baseline, MTL-CPL-U, and MTL-CPL-A. MTL-Baseline does not include any additional layers to incorporate camera model equations, instead, it is an end-to-end learning architecture based on mean absolute error (MAE). It has 13 regressors sharing a common feature extractor directly predicting the required values. MTL-Baseline is implemented to study the effect of proposed camera projection loss. We also used two variants of camera projection loss one with uniform weighting (MTL-CPL-U) in the loss function and the other with adaptive weighting (MTL-CPL-A) to balance the heterogeneous ranges of calibration parameters.

Error Analysis on generated data: We compare the normalized mean absolute error (NMAE) of each of the parameters by all the methods with our proposed approach. It can be seen from Table 1 that for baseline (b) and translation (t_x, t_y, t_z), MTL-Baseline approach resulted in minimum values for NMAE while for all other parameters MTL-CPL-A performs better due to bias in loss introduced as a result of the heterogeneous range of values among parameters. For all the parameters, MTL based methods have lowest NMAE values. This indicates that incorporating camera model geometry in the learning framework not only resulted in a more interpretable learning framework but it also outperforms the state-of-the-art methods.

Error Analysis on real data: For this experiment, we didn't trained on the Tsinghua-Daimler dataset but just performed a forward pass to test the generalizability and the results further strengthen our argument. It can be seen from Table 2 that our proposed multi-task learning approach, MTL-CPL-U, outperforms other methods on 8 out of 10 parameters. For disparity (d), pitch (θ_p) , MTL-CPL-U resulted in higher NMAE values due to bias in loss introduced as a result of the heterogeneous range of values among parameters. For all the remaining parameters our proposed multi-task learning approach resulted in minimum values for NMAE which further solidifies our argument of incorporating camera model geometry in the learning framework.

4. CONCLUSION

Our proposed approach outperforms several baselines, including CNN-based methods on both synthetic and real data is an amalgam of Deep Learning and closed-form analytical solutions. Thus this paper offers a significantly new idea in the area of Machine Learning and Computer Vision. Although we have applied the proposed solution for the estimation of camera calibration parameters, it offers a general framework for many such problems. For example, tracking using Kalman Filter, homography estimation, and many other applications we plan to implement in future.

5. REFERENCES

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun, "CARLA: An open urban driving simulator," in *1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[2] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[3] Scott Workman, Connor Greenwell, Menghua Zhai, Ryan Baltenberger, and Nathan Jacobs, "Deepfocal: A method for direct focal length estimation," in *IEEE International Conference on Image Processing*, 2015, pp. 1369–1373.

[4] Jiangpeng Rong, Shiyao Huang, Zeyu Shang, and Xianghua Ying, "Radial lens distortion correction using convolutional neural networks trained with synthesized images," in *Asian Conference on Computer Vision*. Springer, 2016, pp. 35–49.

[5] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Jonathan Eisenmann, Matthew Fisher, Emiliano Gambaretto, Sunil Hadap, and Jean-François Lalonde, "A perceptual measure for deep single image camera calibration," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2354–2363.

[6] Manuel Lopez, Roger Mari, Pau Gargallo, Yubin Kuang, Javier Gonzalez-Jimenez, and Gloria Haro, "Deep single image camera calibration with radial distortion," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11817–11825.

[7] Menghua Zhai, Scott Workman, and Nathan Jacobs, "Detecting vanishing points using global image context in a non-manhattan world," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5657–5665.

[8] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.

[9] Oleksandr Bogdan, Viktor Eckstein, Francois Rameau, and Jean-Charles Bazin, "DeepCalib: a deep learning approach for automatic intrinsic calibration of wide field-of-view cameras," in *ACM SIGGRAPH European Conference on Visual Media Production*, 2018, pp. 1–10.

[10] Chaoning Zhang, Francois Rameau, Junsik Kim, Dawit Mureja Argaw, Jean-Charles Bazin, and In So Kweon, "DeepPTZ: Deep self-calibration for ptz cameras," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1041–1049.

[11] Scott Workman, Menghua Zhai, and Nathan Jacobs, "Horizon lines in the wild," *arXiv preprint arXiv:1604.02129*, 2016.

[12] João P Barreto, "A unifying geometric representation for central projection systems," *Computer Vision and Image Understanding*, vol. 103, no. 3, pp. 208–217, 2006.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[14] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[15] Olivier Faugeras and Olivier Autor Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT press, 1993.

[16] Xiaofei Li, Fabian Flohr, Yue Yang, Hui Xiong, Markus Braun, Shuyue Pan, Keqiang Li, and Dariu M Gavrila, "A new benchmark for vision-based cyclist detection," in *IEEE Intelligent Vehicles Symposium*, 2016, pp. 1028–1033.

[17] Nikhil Ketkar, "Introduction to Keras," in *Deep learning with Python*, pp. 97–111. Springer, 2017.

[18] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, San Diego, CA, USA, 2015.