Dynamic Intrinsic Parameter Rectification Network for Cameras with Optical Image Stabilization in Desktop Augmented Reality Applications

Shuangyu Xie, Di Wang, Shu-Hao Yeh, Wei Yan, and Dezhen Song

***Abstract*— Optical Image Stabilization (OIS) system in mobile devices reduces image blur by steering lens to compensate for hand jitters. However, the OIS dynamically changes the intrinsic camera matrix, affecting the accuracy of the camera pose or the 3D reconstruction. Here, we propose a novel multilayer perceptron-based approach that estimates the intrinsic matrix in real time. Our network design takes the gridified projection model discrepancy feature and 3D point positions as input to approximate the intrinsic manifold. We also design a unique training scheme for this network by introducing a backpropagated perspective-n-point layer so that the reprojection error can be adopted as the loss function. The training process utilizes precise calibration patterns, but the trained network can be used anywhere. We name the proposed Dynamic Intrinsic Manifold Estimation network as DIME-Net and implement and test it with three different mobile devices. DIME-Net can reduce the reprojection error by at least 71% and the 3D reconstruction error by 53%, indicating that our design is successful.**

I. Introduction

Fig. 1a illustrates how typical OIS functions. When a camera on a mobile device looks at a point in the scene and the camera-held hand jitters, the image blurs because the point is imaged as a short trajectory instead of a point. To mitigate this effect, the camera is often equipped with a motion sensor to sense hand/camera motion. The sensed motion is used to generate a countering motion to actuate camera lens or part of lens array so that the imaged point remains at the same 2D location in the imaging sensor and the stationary part of image remains sharp.

Unfortunately, OIS dynamically changes the intrinsics of the camera, making it difficult to accurately estimate the pose of the camera or perform scene reconstruction. Such applications are often seen in Augmented Reality (AR), which is enabled by handheld mobile devices. The existing practice opts to reduce the resolution of the camera so that the intrinsic matrix K can be approximated by an averaged value, which clearly sacrifices accuracy.

It is worth noting that existing calibration methods cannot be directly applied to our problem because 1) camera intrinsics are dynamic, which break the assumption of constant intrinsics in calibration, and 2) there may not be enough corresponding features in a single frame to recover intrinsics accurately. Therefore, we propose a novel multilayer perceptron (MLP)-based approach to predict the matrix K for desktop AR applications where camera poses are often estimated from 2D and 3D point correspondences, which is just the Perspective-n-Point (PnP) problem [1]. Our network estimates K in real time to cope with the dynamic nature of K under the OIS effect. As a result, downstream tasks, such as camera pose estimation and scene reconstruction algorithms, can achieve higher accuracy. For network input, we propose a gridified project model discrepancy feature. We also design a unique training scheme for this network by introducing a backpropagated PnP (BPnP) layer [2] so that the reprojection error can be adopted as a loss function. Training can be done using precise calibration patterns in laboratory settings, which builds manifold approximation using carefully collected data to ensure good knowledge embedding. Network inference does not require a large number of high-quality features and can be applied to natural objects. We name the dynamic intrinsic manifold estimate network as DIME-Net and have implemented and tested it on three different mobile devices. DIME-Net can reduce both the reprojection error and the 3D reconstruction error by at least 71% and 53%, respectively, indicating that our design is successful.

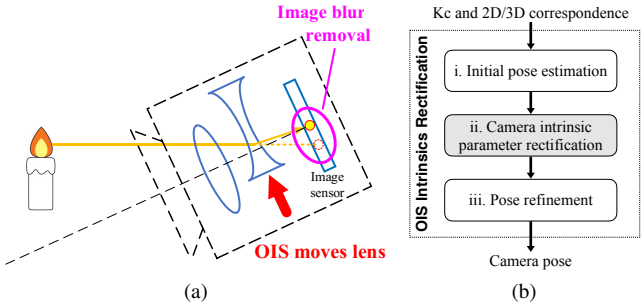


Fig. 1: (a) Illustration of OIS working principle. (b) Pipeline of the OIS intrinsic rectification algorithm.

II. Related Work

In a nutshell, our approach is to train a neural network to track dynamic intrinsic camera parameters. It is related to camera projection modeling, calibration, and a geometry-guided neural network.

The perspective projection model [3], [4], also known as the pinhole model, is the most widely adopted camera

S. Xie, D. Wang, and S. Yeh are co-first authors of this paper. S. Xie, D. Wang, and D. Song are with Department of Computer Science and Engineering, Texas A&M University. D. Song is also with Department of Robotics, Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI) in Abu Dhabi, UAE. Corresponding author: Dezhen Song. Email: dezhen.song@mbzuai.ac.ae

This work was supported in part by National Science Foundation under IIS-2119549 and NRI-1925037, and by GM/SAE Autodrive Challenge II.

model for intrinsics. For a fixed lens camera, its intrinsics are a constant matrix. Of course, this is not true for an OIS-activated camera. Cameras with alterable optical configurations such as a telephoto lens exist and have variable intrinsics [5]–[7]. To model this type of camera, control parameters of optical settings become inputs to intrinsic functions. Similarly, camera developers can obtain OIS servo actuator measurement to estimate intrinsics, such as CIP-VMobile [8]. For most cameras in mobile devices, manufacturers often do not provide lens motion measurement. These factors make it difficult to directly model intrinsics as a function of lens motion, so we have to resort to a hardware-independent approach.

If we know the geometry property of the observed object, we can recover all camera parameters using an estimation method. This is often known as camera calibration. Such methods require a large number of features and are often assisted with carefully designed calibration patterns [9]–[13] to increase accuracy. Among existing calibration methods, self-calibration [14] (or auto-calibration) does not rely on calibration pattern, and finds the camera intrinsics through projective geometry properties existing in image sequences (e.g. absolute conic [3]). However, all of these methods assume constant intrinsics.

Recent research explores the transfer of knowledge in the geometry domain into the design of network architecture for geometry-related problems such as pose estimation [15]–[17] and 3D reconstruction [18], [19]. In the estimation of camera poses, PoseNet [15], [16] uses CNNs and fully connected layers to solve camera pose regression. These works focus on camera extrinsics. Although different from intrinsic parameter estimation, their methods that enforce geometric constraints (e.g., reprojection error [16] along with positional and orientational error [15], [20]) into the loss function for training shed light on how to approach our problem. We employ the reprojection error as a loss function, which is enabled by building on the recent progress on the PnP [1], [2], [21] problem. BPnP [2] considers optimization as a layer and allows for backpropagation of the network as a whole with the help of the implicit theorem [22]. This eventually enables us to employ the reprojection error as our loss function in the DIME-Net training design.

In our design, DIME-Net uses MLP to approximate the manifold that characterizes dynamic intrinsics. There are existing methods that use the learning-based approach for the approximation of the manifold and distance field [23], [24]. Specifically, a previously made effort to use MLP to represent the manifold field. For example, Moser Flow [25] uses MLP to represent a geometry manifold such as Torus. Pose-NDF [26] designs a generative model with an implicit geometry function representing as a feature to create a human pose sequence in a manifold.

III. Problem Definition

Let us first introduce the imaging process and analyze why the existing OIS effect mitigation scheme is problematic before defining the problem.

A. Perspective Projection under OIS

Frequently used coordinate systems and variables are defined as follows.

- $\{C\}$ 3D camera coordinate system (CCS), where its origin is at the camera center, and its X and Y axes parallel to the horizontal and vertical axes of its image 2D coordinate $\{I\}$, respectively.
- $\{W\}$ is a fixed 3D world coordinate system with its origin at the designated checkerboard's top left vertex and its X and Y axes corresponding to the horizontal and vertical directions of the checkerboard, respectively.
- $\mathbf{x}$ is a 2D point in $\{I\}$ as a homogeneous 3-vector $\mathbf{x} \in \mathbb{P}^2$ which is the 2D projective space.
- $\mathbf{X}$ is a 3-vector describing a 3D point position. By convention, we use left superscript to indicate the reference frames of 3D points (e.g. ${}^{W}\mathbf{X}$ is a point in $\{W\}$).

All 3D coordinate systems are right-handed. For a regular camera that follows the perspective projection model [3], it projects a 3D point ${}^{W}\mathbf{X}$ to a 2D image point $\mathbf{x}$ as follows.

$$\mathbf{x} = \lambda \mathrm{K}[{}^{C}_{W}\mathrm{R}\ {}^{C}_{W}\mathbf{t}] \begin{bmatrix} {}^{W}\mathbf{X} \\ 1 \end{bmatrix}, \tag{1}$$

where λ is a scalar and matrix $\mathrm{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$ is the intrinsic matrix of the camera, f_x and f_y are focal lengths in pixel counts using pixel width and height, respectively, and (c_x, c_y) is principal point location on the image. Note that this is a 4-Degrees of Freedom (DoFs) intrinsic matrix model which fits most cameras. We also call K intrinsics for brevity. Similarly $[{}^{C}_{W}\mathrm{R},\ {}^{C}_{W}\mathbf{t}] \in \mathcal{SE}(3)$, is called extrinsics. If 3D points live in $\{C\}$, then ${}^{C}_{W}\mathrm{R} = \mathrm{I}_3$ becomes an identity matrix and ${}^{C}_{W}\mathbf{t} = \mathbf{0}_3$, a zero vector in $\mathbb{R}^3$.

When the OIS is activated, both the relative orientation and the distance between the lens and the 2D imaging sensor varies dynamically. From Fig. 1a, each element in K is a function of the lens pose $[\mathrm{R}_{\text{lens}}, \mathbf{t}_{\text{lens}}] \in \mathcal{SE}(3)$. Therefore, it can be written in function format $\mathrm{K}(\mathrm{R}_{\text{lens}}, \mathbf{t}_{\text{lens}})$. An immediate thought would be if it is possible to directly model the function $\mathrm{K}(\mathrm{R}_{\text{lens}}, \mathbf{t}_{\text{lens}})$ based on lens motion $(\mathrm{R}_{\text{lens}}, \mathbf{t}_{\text{lens}})$. Unfortunately, it is very difficult to do so due to the lack of information about the design of the OIS system for each mobile device. Depending on how sophisticated an OIS is, its lens may have up to 5 DoFs, although a typical mobile device camera may only has 2 rotational DoFs due to cost and size concerns. The lack of detailed information on the mechanism is not the only issue. Also, we do not have access to the lens motion feedback since most device software development kits (SDKs) do not provide it. Finally, we do not know to which time epoch the OIS aligns frames. These factors determine that modeling OIS is not a viable approach, and we have to opt for a data-driven approach that can be used in a wide range of devices or OIS types.

B. Existing OIS Effect Mitigation Scheme

The dynamic intrinsics immediately lead to a problem for any vision algorithm that requires a constant or known K matrix. More specifically, any 3D reconstruction or camera pose estimation algorithms would be severely impacted. Existing practices adopted by cellphone manufacturers such as Apple™ or Google™ often resort to a prior approximated intrinsics denoted as K_c on reduced resolution images in their SDKs. Such a K_c is often obtained by averaging a large number of K's at different OIS states or at its neutral stationary positions. For K_c, we can associate it with extrinsics which defines a unique camera frame $\{C_0\}$.

In fact, we have also tested the previous K_c in our experiment setup using the PnP problem as an example [1]. Denote the i-th 2 D point by $\mathbf{x}_i$. The 2D-3D point correspondences are defined as $\{\mathbf{x}_i \leftrightarrow {}^{W}\mathbf{X}_i : i = 1, \cdots, n\}$, where n is the total number of point correspondences. PnP algorithm computes camera pose using the 2D-3D correspondences by minimizing reprojection error

$$[{}^{C}_{W}\mathrm{R},\ {}^{C}_{W}\mathbf{t}] = \operatorname{argmin} \sum_i \left\| \lambda_i \mathrm{K} ({}^{C}_{W}\mathrm{R}\, {}^{W}\mathbf{X}_i + {}^{C}_{W}\mathbf{t}) - \mathbf{x}_i \right\|^2_{\Sigma}, \tag{2}$$

where $\|\cdot\|_{\Sigma}$ is the Mahalanobis norm with the covariance matrix Σ for pixel location distribution.

In real-world applications, the average reprojection error would be much higher because the pixelization error from the real scene is much higher than the precise and sharp inputs from the calibration pattern. Higher error would cause the algorithm to struggle to converge under noisy input. Consequently, existing practices are to lower the image resolution to increase the pixel size. This approach is to sacrifice image resolution and camera pose accuracy for algorithm stability, which is not ideal because we cannot fully utilize the true potential of the camera resolution.

C. OIS Intrinsics Rectification Framework

It is necessary to rectify K_c. If a precise K can be obtained in some way in real time, then the problem is solved. Again, let us use PnP as an example, since it is the most typical scenario in desktop AR applications. A quick thought would be if we could add K as an additional decision variable in the estimation problem in (2) to address the problem. Unfortunately, this would not work because the number of point correspondences in an application is usually insufficient or unevenly distributed, which cannot guarantee K's quality. Since we do not have a clear path to estimate K analytically, the idea becomes if we could find a data-driven approach. The general framework is shown in Fig. 1b with three main blocks as follows. The first step (Box i) is the initial estimate of the pose using the previous K_c. We know that this pose estimation will not be accurate enough, but its residual errors are caused by the discrepancy between K_c and the actual K and hence are important input for the next step. The second step (Box ii) is to recover the K and the third step (Box iii) is pose refinement with the newly obtained K which is simply to re-solve the PnP problem with the new K. It is clear that the second step is the key problem here. Let us define this problem,

Definition 1: Given the K_c and n point correspondences $\{\mathbf{x}_i \leftrightarrow {}^{C_0}\mathbf{X}_i\}_{i=1}^{n}$, design and train DIME-Net to represent the f_{K} manifold that can be used to predict the dynamic intrinsic camera matrix K.

Here we assume that non-linear lens distortion has been removed from images. Cameras with OIS usually have non-linear lens distortion removed to facilitate OIS.

IV. DIME-Net Design and Training

The K variation caused by the OIS effect can be considered as a f_{K} manifold despite the fact that we do not have a close form representation of $\mathrm{K}(\mathrm{R}_{\text{lens}}, \mathrm{t}_{\text{lens}})$. In fact, the pose of the lens $[\mathrm{R}_{\text{lens}}, \mathrm{t}_{\text{lens}}]$ is just the extrinsics of the camera $[{}^{C}_{W}\mathrm{R},\ {}^{C}_{W}\mathbf{t}]$ in a different reference system under the actual K. Therefore, we know that these point correspondences have to satisfy (1). On the other hand, the first step of Sec. III-C also produces projected points.

$$\mathbf{x}_c = \lambda_c \mathrm{K}_c \begin{bmatrix} {}^{C_0}_{W}\mathrm{R} & {}^{C_0}_{W}\mathbf{t} \end{bmatrix} \begin{bmatrix} {}^{W}\mathbf{X} \\ 1 \end{bmatrix}, \tag{3}$$

where corresponding variables with subscription c indicate that they are estimated based on K_c. Define $\Delta\mathbf{x} = \mathbf{x}_c - \mathbf{x}$. We know that

$$\Delta\mathbf{x} = \left\{ \lambda_c \mathrm{K}_c \begin{bmatrix} {}^{C_0}_{W}\mathrm{R} & {}^{C_0}_{W}\mathbf{t} \end{bmatrix} - \lambda \mathrm{K} \begin{bmatrix} {}^{C}_{W}\mathrm{R} & {}^{C}_{W}\mathbf{t} \end{bmatrix} \right\} \begin{bmatrix} {}^{W}\mathbf{X} \\ 1 \end{bmatrix}. \tag{4}$$

With the same point correspondences, keep in mind that the extrinsics $[{}^{C_0}_{W}\mathrm{R},\ {}^{C_0}_{W}\mathbf{t}]$ and $[{}^{C}_{W}\mathrm{R},\ {}^{C}_{W}\mathbf{t}]$ are functions of the corresponding intrinsics K_c and K, respectively. This means that (4) defines an input-dependent f_{K} manifold:

$$f_{\mathrm{K}}(\mathrm{K}, \mathrm{K}_c, \{\Delta\mathbf{x}_i, \mathbf{X}_i, \forall i\}) = 0. \tag{5}$$

It is not difficult to see that f_{K} becomes less dependent on the individual $\{\mathbf{x}_i, \mathbf{X}_i\}$ as i grows large. At this stage, f_{K} can be used to predict K for a small number of correspondences. This inspires us to develop a data-driven approach to represent the f_{K} manifold using our DIME-Net. Construction of the approximated vector field f_{K}, K_c and $\{\Delta\mathbf{x}_i, \mathbf{X}_i, \forall i\}$, mapping from the input feature vector to the dynamic K is the DIME-Net training process. It can be done with carefully collected data under different OIS states with calibration patterns under laboratory settings. Later in the application, this DIME-Net can be used as a K predictor.

Fig. 2 shows our DIME-Net architecture. We first design the input for DIME-Net which converts the point correspondences and the prior camera matrix into the 1D OIS discrepancy feature. Given the 1D OIS discrepancy feature, DIME-Net utilizes the MLP to rectify the intrinsics. We will explain how we design DIME-Net with its unique feature, network structure, and network loss function.

A. OIS Discrepancy Feature

Denote the 3D position ${}^{C_0}\mathbf{X}_i := [X_i, Y_i, Z_i]^{\mathsf{T}} \in \mathbb{R}^3$ and the corresponding 2D pixel position $\tilde{\mathbf{x}}_i := [x_i, y_i]^{\mathsf{T}} \in \mathbb{R}^2$ where symbol ˜ on a variable means that it is in inhomogeneous coordinate. Note that 3D points are in $\{C_0\}$ that

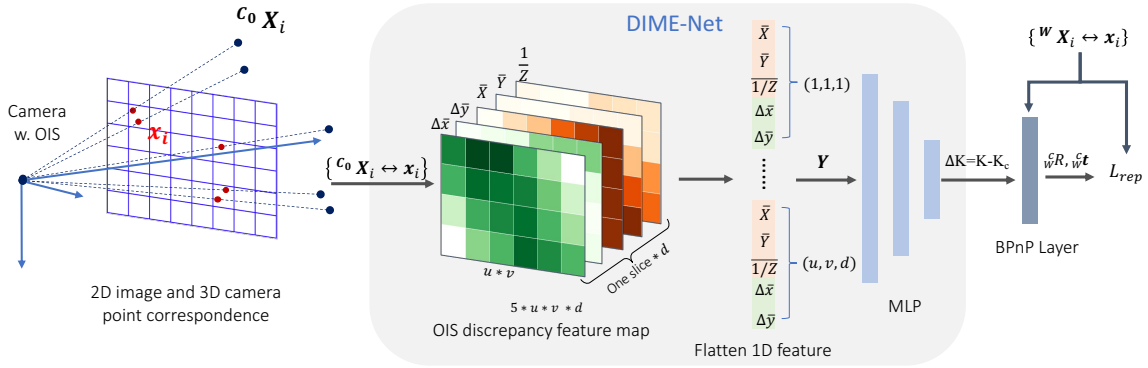


Fig. 2: DIME-Net architecture and training scheme. This pipeline reflects the process of training the DIME-Net. In the inference stage, the user only needs the gray box to estimate K and the pose can be calculated using the standard PnP algorithm [1]. For illustration purposes, the 3D feature map only shows one slice.

the manifold will be defined in $\{C_0\}$ instead of $\{W\}$ as (4). This change makes the neural network not sensitive to the choice of world coordinate system. There are 3 steps to obtain the input feature of the neural network: (1) point-based OIS discrepancy feature conversion, (2) grid-based OIS discrepancy feature conversion and (3) 1D OIS discrepancy feature flattening.

1) Point-Based OIS Discrepancy Feature: Each point-based OIS discrepancy feature is composed of two main components: (1) the inhomogeneous representation of $\Delta\mathbf{x}$ which is named the projection model discrepancy feature (PMD) because it is the 2D reprojection error between the observed image points and their re-projected points using K_c, and (2) 3D point positions in $\{C_0\}$. The PMD is the direct OIS effect on the K [27] when the 3D point position is given in $\{C_0\}$.

Denote the PMD of $\mathbf{x}_i \leftrightarrow {}^{C_0}\mathbf{X}_i$ as $\left[\Delta x_i, \Delta y_i\right]^\mathsf{T} \in \mathbb{R}^2$

$$\begin{bmatrix}\Delta x_i \\ \Delta y_i\end{bmatrix} = \begin{bmatrix}\mathbf{k}_c^1 \\ \mathbf{k}_c^2\end{bmatrix} \begin{bmatrix}X_i/Z_i \\ Y_i/Z_i \\ 1\end{bmatrix} - \begin{bmatrix}x_i \\ y_i\end{bmatrix}, \tag{6}$$

where $\mathbf{k}_c^j$ is the j-th row of K_c. It is worth noting that (6) is the simplification of (4) in $\{C_0\}$. We then concatenate the PMD and the 3D point position in $\{C_0\}$ to form the point-based OIS discrepancy feature. Denote the point-based OIS discrepancy feature of $\mathbf{x}_i \leftrightarrow {}^{C_0}\mathbf{X}_i$ by $\mathbf{f}_i$ which is defined as

$$\mathbf{f}_i := \left[\Delta x_i, \Delta y_i, X_i, Y_i, 1/Z_i\right]^\mathsf{T} \in \mathbb{R}^5. \tag{7}$$

It is worth noting that the inverse depth $1/Z_i$ is used since the inverse depth is linear to $\mathbf{k}_c^1$ and $\mathbf{k}_c^2$ in (6) making the model more linear and fits better to the neural network.

2) Grid-Based OIS 3D Feature Map: However, there are two remaining issues when using point-based features: 1) the point-based feature number is not fixed because it is input-dependent, and 2) the order of features should be irrelevant. In fact, the features should be related to the 2D position in the image. If we blindly feed the point-based features into a neural network, we would run into issues because 1) the neural network would need a fixed input dimension and 2) the neural network would inevitable learn the order of inputs instead of the spatial location in the image. To address these issues, we convert point-based features into grid-based features by using grid cells and merging feature information within each grid cell. This approach fixes the input dimension and order issues since there is a constant number of grid cells, and we can arrange the grid feature using the lexicographic order of cells.

First, we create a 2D feature map which has the same size as the original image but with 5 channels that contains a point-based OIS discrepancy feature $\mathbf{f}_i$ for each point correspondence. Next, we divide the 2D point-based feature map into a 2D grid pattern consisting of $u \times v$ equal-sized square grid cells. To further encode the depth information to the network and reduce the network's sensitivity to 3D noise, we discretize the depth space as the third dimension into d slices, and the feature map becomes 3D. For each cell in the 3D feature map, we average the point-based features in the cell to be the corresponding grid-based features. The feature is indexed by its 2D point position (x_i, y_i) and its depth Z_i on the 3D feature map. Let $\left[a_k, b_j, c_l\right]^\mathsf{T}$ be the bottom-right corner point pixel position of the grid in the j-th row, the k-th column, l-th slice of the 3D grid pattern. The point-based feature set that reside in the grid in the j-th row, the k-th column, and the l-th slice is defined as

$$\mathcal{F}_{j,k,l} := \left\{\mathbf{f}_i : x_i \in [a_{k-1}, a_k) \ \wedge \ y_i \in [b_{j-1}, b_j) \wedge Z_i \in [c_{l-1}, c_l)\right\}. \tag{8}$$

Denote the grid-based OIS discrepancy feature of the grid in the j-th row, the k-th column and l-th slice as $\mathbf{y}_{j,k,l}$. The grid-based OIS discrepancy feature $\mathbf{y}_{j,k,l}$ is defined as

$$\begin{aligned}\mathbf{y}_{j,k,l} :=& \frac{1}{|\mathcal{F}_{j,k,l}|} \sum_{\mathbf{f}_i \in \mathcal{F}_{j,k,l}} \mathbf{f}_i \\ =& \left[\Delta\overline{x}_{j,k,l}, \Delta\overline{y}_{j,k,l}, \overline{X}_{j,k,l}, \overline{Y}_{j,k,l}, \overline{1/Z}_{j,k,l}\right]^\mathsf{T} \in \mathbb{R}^5,\end{aligned} \tag{9}$$

where $|\cdot|$ is the set cardinality and symbol $\bar{}$ on a variable indicates the average value.

3) 1D OIS Discrepancy Feature Flattening:: We flatten the grid-based features to one dimension as the final input for the MLP in the next step. Denote the flattened feature vector

as $\mathbf{y}$. The flattened vector $\mathbf{y}$ can be obtained by concatenating the grid-based features,

$$\mathbf{y} := \left[\mathbf{y}_{1,1,1}^{\mathsf{T}}, \mathbf{y}_{1,1,2}^{\mathsf{T}}, \ \cdots, \ \mathbf{y}_{1,2,1}^{\mathsf{T}}, \ \cdots, \ \mathbf{y}_{u,v,d}^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{m_y}, \quad (10)$$

where u, v and d are the numbers of the grid cells in row, column and slice, respectively, and dimension $m_y = 5uvd$.

B. DIME-Net Architecture and Loss Function

Fig. 2 shows that we employ a multilayer perceptron network to learn the f_{K} manifold to generate dynamic K from $\mathbf{y}$. We employ geometric error as a loss function to link the network's performance to the camera projection model.

1) Multilayer Perceptron: Recall that $\mathbf{y}$ is the feature vector of the point correspondence that describes the OIS effect. Our goal is to generate K from $\mathbf{y}$. From the perspective of the OIS feature, the intrinsic value K is a latent variable that directly describes the camera model. We design an MLP to be an antoencoder-style mapping from a high-dimensional feature variable $\mathbf{y}$ to a low-dimensional latent variable K. Specifically, we employ a fully connected 3-layer perceptron to generate K. We design the network output to be $\Delta\mathrm{K} = \mathrm{K} - \mathrm{K}_c$. The output layer has 4 nodes that represent four components of ΔK: Δf_x, Δf_y, Δc_x, and Δc_y. This design helps regulate the network. In the special case when input vector $\mathbf{y} = \mathbf{0}_{m_y}$, the network should output $\mathbf{0}_4$ so that $\Delta\mathrm{K} = 0_{3\times3}$ and $\mathrm{K} = \mathrm{K}_c$ due to lack of information, which also ensures the stability of the network.

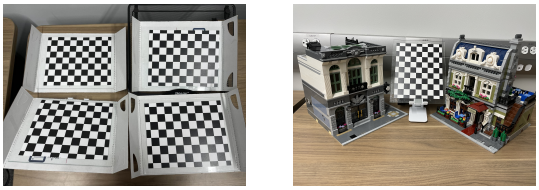
2) BPnP Layer in Training and Loss Function Design: For the network training, we use the reprojection error as a loss function. This directly ties network performance to model quality. Given the predicted intrinsics K and extrinsics ${}^C_W\mathrm{R}$ and ${}^C_W\mathbf{t}$ and the point correspondences $\{\mathbf{x}_i \leftrightarrow {}^W\mathbf{X}_i\}$, the loss can be calculated by

$$L_{\text{rep}} = \sum_i \left\| \lambda_i \mathrm{K}({}^C_W\mathrm{R}\,{}^W\mathbf{X}_i + {}^C_W\mathbf{t}) - \mathbf{x}_i \right\|^2_\Sigma. \quad (11)$$

Note that we would need extrinsics $[{}^C_W\mathrm{R}, {}^C_W\mathbf{t}]$ to compute the loss function. To obtain extrinsics and enable end-to-end training of the network, we connect the network with a BPnP layer [2] to estimate $[{}^C_W\mathrm{R}, {}^C_W\mathbf{t}]$ from the predicted K. Compared to the general PnP solver, BPnP considers optimization as a layer and enables the backpropagation of the network as a whole with the help of the implicit theorem [22]. Using the reprojection error [3] as our loss function makes the overall model like a maximum likelihood estimator. Common loss functions like the L1 or L2 norm are algebraic distance which is not robust and can lead to a spurious solution since it does not contain the geometric meaning. The reprojection error, on the other hand, is a geometric distance. Therefore, the loss function in (11) can guide the network in learning the f_{K} manifold.

C. Training and Inference Using DIME-Net

To gather good training samples, as shown in Fig. 3a, we have designed a calibration rig. It contains a 4-checkerboard pattern located at four different planes placed on a planar glass to ensure flatness, each of which contains inner vertices of 8×10. The side length of each cell is 22.0 mm. The positions of the 3D points are computed in $\{C_0\}$, and the 2D points are read from the vertex coordinates in the image. It should be noted that the 4-checkerboard rig design allows us to directly obtain K for each image using a calibration procedure because there are enough inputs to estimate both K and extrinsics. This is very important in training and verification because it provides the ground truth. With this setup, we can obtain a set of 2D-3D correspondences with a moving camera at different perspectives that covers the normal working range of the camera.



(a) (b)

Fig. 3: (a) Example image of the training inputs for DIME-Net using the calibration rig. (b) Example image of natural object feature test setup where two LEGO buildings are the natural objects.

The coverage of the training data is designed for desktop AR applications. The configuration ensures that our neural network can approximate f_{K} manifold with good accuracy. With a trained network, our DIME-Net has the ability to predict ΔK given the input feature vector in (10) converted from the 2D-3D point correspondence set.

V. Experiments

We have implemented our DIME-Net[1] using PyTorch [28]. First, we perform an ablation study of our DIME-Net. Then we evaluate the inference accuracy of our DIME-Net using both the calibration rig and the natural object. Let us introduce our OIS datasets.

1) Calibration Rig OIS Datasets: We have collected data under the OIS effect using the aforementioned calibration rig in Sec. IV-C. To activate the OIS, we hold the camera and capture images with different poses. We use three different cameras as detailed in Tab. I. For each camera, we collect a data set and split it into a training set and a testing set (shown as #images in the columns "Train" and "Test").

For each device, we obtain K_c according to the method in Sec. III-C. In addition, for each image, we also use the 4-board as input to estimate $\mathrm{K}^\star$ using the camera calibration method in [29] by solving the following nonlinear optimization,

$$\min_{{}^C_W\mathrm{R}, {}^C_W\mathbf{t}, K} \quad \sum_{i\in N} \left\| \lambda_i \mathrm{K}({}^C_W\mathrm{R}\,{}^W\mathbf{X}_i + {}^C_W\mathbf{t}) - \mathbf{x}_i \right\|^2_\Sigma, \quad (12)$$

where we use all the point correspondence in one image $N = 320$ to obtain the K^*.

[1]Code will be released after review period to respect the conference anonymous rule.

The calibration process method produces a reprojection error $e^\star$. The average reprojection errors are shown in the Avg($e^\star$) column in pixels which provide a baseline for the best possible performance for the reprojection error.

TABLE I: Calibration rig OIS image datasets

Device	Resolution	Train	Test	Avg($e^\star$)
Samsung Galaxy S8	4032×3024	185	47	0.45
iPhone 12 Pro	4032×3024	164	42	0.46
iPad mini 6	4032×3024	224	57	0.32

A. DIME-Net Ablation Study

Next we test the impact of different feature setups for DIME-Net performance using Samsung Galaxy S8 data from Tab. I.

1) Grid Resolution and Occupancy Tests: Now we test how grid resolution and grid cell occupancy can affect DIME-Net performance under reprojection error e. Define the average reprojection error as Avg(e) which is used as a primary metric. The resolution of the grid $u \times v \times d$ determines the number of points-based OIS discrepancies in each cell and affects the uncertainty of the points-based OIS discrepancies, which are the direct input of the DIME-Net. The grid occupancy, on the other hand, indicates the distribution of the preserved OIS information. As shown in Tab. II, we have chosen 4 different grid resolution: $16 \times 12 \times 1$, $12 \times 9 \times 1$, $8 \times 6 \times 1$ and $8 \times 6 \times 3$. To simulate occupancy, we uniformly sample cells and empty the 2D and 3D point correspondences in the cells. The empty cell ratio is measured by $\eta = 1 - \frac{m'_p}{m_p}$, where m_p and m'_p are the number of cells with non-zero OIS features before and after sampling, respectively. The grid occupancy then is measured by $\gamma = \frac{m_p}{uvd}$.

TABLE II: Avg(e) in pixels vs. grid resolution and occupancy in different η and γ. Smaller is better. Best results are in boldface.

	Grid resolution							
	$16 \times 12 \times 1$		$12 \times 9 \times 1$		$8 \times 6 \times 1$		$8 \times 6 \times 3$	
η%	γ%	Avg(e)	γ%	Avg(e)	γ%	Avg(e)	γ%	Avg(e)
0	64.5	0.73	78.6	0.75	96.4	**0.68**	24.2	0.77
20	52.1	0.87	62.6	0.97	79.3	0.92	23.9	0.77
40	38.9	1.29	46.0	1.33	57.7	1.45	23.6	0.80
60	26.6	1.77	30.5	1.88	36.6	2.05	22.0	0.85
80	13.4	2.47	14.8	2.49	19.7	2.52	16.9	1.10

Tab. II shows that the grid resolution with $8 \times 6 \times 1$ can achieve the lowest Avg(e). The size of the cell and the number of point-based OIS discrepancy features are expected to increase as the grid resolution reduces. The average in (9) reduces the noise of the features when there are more points in each cell. The lowest Avg(e) of 0.68 pixels is close to the calibration accuracy of Avg($e^\star$) = 0.45 in Tab. I which confirms that our DIME-Net works effectively in learning the f_{K} manifold. Meanwhile, the $8 \times 6 \times 3$ setup shows high stability during point reduction because the explicit depth feature makes the model more robust, and the initial model is trained on low density, so reducing the point does not affect the performance. This is a more preferable setup in the application. The results show the effective design of the DIME-Net feature because it is capable of predicting intrinsics even when the grid occupancy is extremely low.

2) OIS Discrepancy Feature Tests: Next, we examine the components of the OIS discrepancy feature in (6). Again, Avg(e) in pixels is used as a metric. We choose $8 \times 6 \times 1$ as grid resolution. We compare five different setups.

A. Complete OIS discrepancy feature in (9) using both PMD and 3D point positions.
B. Only use PMD in (6).
C. Combine PMD with inverse depth $1/Z$.
D. Similar to "C", but we combine PMD with X and Y positions of 3D points.
E. Only use 3D point positions.

Tab. III shows that option A achieves the lowest Avg(e) which means that all features are necessary to achieve the best result. This is not surprising since (4) has told us that. What is interesting is that the performance of options B-D is slightly worse than that of A, which indicates that PMD is the dominating feature.

TABLE III: Avg(e) in pixels under different feature combination

	A	B	C	D	E
Avg(e)	**0.68**	0.78	0.78	0.75	2.58

B. Inference Accuracy Comparison

After knowing the best setup for DIME-Net, we are ready to test its accuracy in inference test. Initially, we use Avg(e) as basic performance metric and we will add the 3D reconstruction error Avg(e_{3D}) later in the comparison for natural objects.

From Sec. III-B, we know that the popular existing approach is to employ the prior K_c that is obtained when the camera is stationary or by averaging a large number of K's under different OIS states. Let us define Avg(e_c) as its average reprojection error when only using K_c.

We also set a baseline for comparison. The baseline is characterized by $\mathrm{K}^\star$ which is the best intrinsics that can be obtained for the test case. Recall that Avg($e^\star$) is its reprojection error. Avg($e^\star$) reflects noises in pixels which is the level of noise that cannot be canceled by adjusting the intrinsics without overfitting.

It is not difficult to see that Avg($e^\star$) $\leq$ Avg(e_c) given a reasonable large population of point correspondences. It is also clear that if our design is effective, then Avg(e) should fall between the two. The closer Avg(e) is to Avg($e^\star$), the better it is. This can be measured by a new metric: the average reprojection error reduction ratio, $\rho = \frac{\mathrm{Avg}(e_c) - \mathrm{Avg}(e)}{\mathrm{Avg}(e_c) - \mathrm{Avg}(e^\star)}$. Higher ρ is more desirable. Now we are ready to compare the inference quality in different datasets.

1) Calibration Rig Inference Accuracy Tests: The first test is performed with the data shown in Tab. I based on the calibration rig data.

2) Point Dropping and Noise Injection Tests: We want to test the inference accuracy of DIME-Net after we decrease the number of point correspondences and/or inject noise to 2D and 3D points. This is important because real-world

applications do not always have ample features at the calibration board point accuracy. To generate the testing condition of decreased point numbers, we uniformly sample the point correspondences to be dropped. For noise injection, we inject random zero-mean Gaussian noise with standard deviation σ_x^+ and σ_X^+ into the 2D point and the 3D point, respectively. It should be noted that the injected noise σ_x^+ and σ_X^+ are the additional noise added to the checkerboard vertices. The units of σ_x^+ and σ_X^+ are pixels and mm, respectively. In this test, we used the Samsung Galaxy S8 camera with a grid resolution of $8 \times 6 \times 3$ for DIME-Net.

Tab. IV shows the results. Note that in our experimental setup 1 mm means about 5.45 pixels (px). The upper half of the table is the result when zero injected noise is added (σ_x^+=0 px , σ_X^+=0 mm), and the lower half of the table is the result when $\sigma_x^+ = 3$ px and $\sigma_X^+ = 0.15$ mm. The average ratio of reduction of the reprojection error, ρ, shows that our DIME-Net is insensitive to injected noise and the low number of point correspondences. It remains close to or above 84% in all settings. Our design has been shown to be effective and robust against point dropping and noisy inputs.

TABLE IV: Inference accuracy measured by average reprojection error in pixels and reduction ratio under different sample sizes: upper half setting is $\sigma_x^+ = 0$ px & $\sigma_X^+ = 0$ mm, and lower half setting is $\sigma_x^+ = 3$ px & $\sigma_X^+ = 0.15$ mm.

#Samples	Avg(e_c)	Avg(e)	Avg($e^\star$)	ρ(%)
320	3.44	0.77	0.45	**89.3**
256	3.34	0.77	0.45	87.9
192	3.43	0.80	0.45	88.3
128	3.45	0.85	0.46	87.0
64	3.40	0.93	0.45	84.0
320	5.28	3.34	3.16	**91.5**
256	5.27	3.36	3.14	89.7
192	5.28	3.37	3.13	88.8
128	5.31	3.42	3.11	85.6
64	5.56	3.46	3.14	86.7

3) Multi-device tests:: We repeat the tests for all three devices using data in Tab. I with the same settings as in the upper half of Tab. IV. The results in the upper half of Tab. V are consistent with previous tests: our DIME-Net achieves over 91% in ρ in all cases.

TABLE V: Inference accuracy measured by the average reprojection error in pixels and reduction ratio. The Scene-Device column beginning with B indicates the results from the checkerboard-based tests, while those beginning with N indicate the results from the natural object-based tests. S8 stands for Samsung Galaxy S8.

Scene-Device	Avg(e_c)	Avg(e)	Avg($e^\star$)	ρ(%)
B-Samsung Galaxy S8	3.44	0.68	0.45	91.7
B-iPhone 12 pro	2.36	0.61	0.46	92.3
B-iPad mini 6	1.61	0.36	0.32	**96.8**
N-S8 (8×6×1)	4.26	3.74	3.21	49.5
N-S8 (8×6×3)	4.26	3.51	3.21	**71.7**

4) Natural Object Inference Accuracy Tests: We also perform an inference accuracy comparison test on natural objects, since this indicates its performance in real-world scenarios where the points are not from the precise calibration pattern. Here, we employ two LEGO buildings as natural objects (see Fig. 3b). We manually select/label the features of the LEGO buildings to ensure perfect correspondence, where there are 47 features on the left side LEGO building and 19 features on the right side LEGO building. Using LEGO models allows us to have 3D points from their CAD models as ground truth. In fact, the ground-truth 3D points are the collection of the 3D natural points and those from the checkerboard in the middle. We have collected 34 images with different poses with a handheld camera to activate OIS during image capture. For network setup, we chose grid resolutions of $8 \times 6 \times 1$ and $8 \times 6 \times 3$ because they have performed better than other configurations in previous tests. We first have used the calibration rig dataset trained network to predict K from the 34 images. The last two rows of Tab. V show that DIME-Net has achieved at least 71% in reducing the reprojection error.

In addition to the previous comparison based on the reprojection error, we have employed the 3D reconstruction error, since the former may be biased by overfitting. We have performed a 3D reconstruction of the entire scene using the network output K and compared it with K_c and $\mathrm{K}^\star$. From the collected images, we have selected two-view pairs with at least 64 point correspondences as the test data, which result in 289 image pairs. We have conducted two-view-based 3D reconstruction on them. Specifically, we first use all point pairs to solve the fundamental matrix. Then the relative pose of the cameras can be decomposed from the two-view intrinsics. Based on the poses and intrinsics, the 3D points are obtained from the triangulation. We estimate a similarity transformation by aligning the checkerboard points from the reconstruction to their counterparts in $\{W\}$. The similarity transformation is then used to transfer all n points on the natural objects to $\{W\}$ as ${}^W\mathbf{X}_i$ for comparison purposes. The average 3D reconstruction error $\mathrm{Avg}(e_{3\mathrm{D}})$ is calculated as $\mathrm{Avg}(e_{3\mathrm{D}}) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\|{}^W\bar{\mathbf{X}}_i - {}^W\mathbf{X}_i\|_2^2}$, where ${}^W\bar{\mathbf{X}}_i$ is the ground truth points. Recall that $\mathrm{Avg}(e_{3\mathrm{D}})$ is calculated from the rectified network K, $\mathrm{Avg}(e_{3\mathrm{D}}^c)$ from K_c, and $\mathrm{Avg}(e_{3\mathrm{D}}^\star)$) from $\mathrm{K}^\star$. We define the average reduction ratio for the reconstruction error as $\rho_{3\mathrm{D}} = \frac{\mathrm{Avg}(e_{3\mathrm{D}}^c) - \mathrm{Avg}(e_{3\mathrm{D}})}{\mathrm{Avg}(e_{3\mathrm{D}}^c) - \mathrm{Avg}(e_{3\mathrm{D}}^\star)}$. It is worth noting that we only rely on points from the LEGO buildings for K inference, and checkerboard points are used as part of the validation for the full scene reconstruction. Tab. VI shows the results in the 3D construction. DIME-Net has achieved 53% reductions in the 3D reconstruction error. In fact, the accuracy improvement results from natural object feature-based inference also show that our training scheme design using the calibration rig successfully learns the f_{K} manifold, since the setups between the LEGO OIS dataset and the calibration rig OIS dataset are very different.

TABLE VI: Average 3D reconstruction error in mm.

Scene-Device	Avg($e_{3\mathrm{D}}^c$)	Avg($e_{3\mathrm{D}}$)	Avg($e_{3\mathrm{D}}^\star$)	$\rho_{3\mathrm{D}}$(%)
N-S8 (8×6×1)	8.63	7.61	5.48	32%
N-S8 (8×6×3)	8.63	6.94	5.48	**53**%

VI. Conclusions and Future Work

To deal with the camera intrinsics variation caused by OIS system, we presented our new DIME-Net, a multilayer

perceptron network designed to rectify camera intrinsics in real time. We analyzed the OIS system and proposed to use a gridified PMD feature set along with 3D point positions to train DIME-Net using calibration patterns. The trained network became an approximation of the intrinsics manifold that can predict rectified intrinsics in desktop AR applications. We have implemented and extensively tested our design. The experimental results confirmed that our design was robust and effective and can significantly reduce the reprojection error and the 3D reconstruction error. In the future, we will improve our design with better geometry insights for more applications.

Acknowledgment

We thank Y. Xu and Z. Shaghaghian for their insightful discussions. We are also grateful to A. Kingery, F. Guo, C. Qian, and Y. Jiang for their inputs and feedback.

References

[1] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009. 1, 2, 3, 4

[2] B. Chen, A. Parra, J. Cao, N. Li, and T.-J. Chin, "End-to-end learnable geometric vision by backpropagating pnp optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8100–8109. 1, 2, 5

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge Univ Pr, 2003. 1, 2, 5

[4] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006. 1

[5] J.-M. Lavest, G. Rives, and M. Dhome, "Three-dimensional reconstruction by zooming," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 2, pp. 196–207, 1993. 2

[6] G. Simon and M.-O. Berger, "Registration with a zoom lens camera for augmented reality applications," in *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. IEEE, 1999, pp. 103–112. 2

[7] J. Miura, T. Kanda, and Y. Shirai, "An active vision system for real-time traffic sign recognition," in *ITSC2000. 2000 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 00TH8493)*. IEEE, 2000, pp. 52–57. 2

[8] L. Jin, H. Zhang, and C. Ye, "Camera intrinsic parameters estimation by visual–inertial odometry for a mobile phone with application to assisted navigation," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 1803–1811, 2020. 2

[9] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987. 2

[10] R. G. Willson, "Modeling and calibration of automated zoom lenses," in *Videometrics III*, vol. 2350. SPIE, 1994, pp. 170–186. 2

[11] M. Li and J.-M. Lavest, "Some aspects of zoom lens camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 11, pp. 1105–1110, 1996. 2

[12] J. Heikkila and O. Silvén, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1997, pp. 1106–1112. 2

[13] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000. 2

[14] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank, "Camera self-calibration: Theory and experiments," in *European conference on computer vision*. Springer, 1992, pp. 321–334. 2

[15] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946. 2

[16] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5974–5983. 2

[17] C. M. Parameshwara, G. Hari, C. Fermüller, N. J. Sanket, and Y. Aloimonos, "Diffposenet: Direct differentiable camera pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 6845–6854. 2

[18] A. Boulch and R. Marlet, "Poco: Point convolution for surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 6302–6314. 2

[19] Y. Ding, W. Yuan, Q. Zhu, H. Zhang, X. Liu, Y. Wang, and X. Liu, "Transmvsnet: Global context-aware multi-view stereo network with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8585–8594. 2

[20] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-aware learning of maps for camera localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2616–2625. 2

[21] D. Campbell*, L. Liu*, and S. Gould, "Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization," in *ECCV*, 2020, * equal contribution. 2

[22] S. G. Krantz and H. R. Parks, *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002. 2, 5

[23] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 597–613. 2

[24] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020. 2

[25] N. Rozen, A. Grover, M. Nickel, and Y. Lipman, "Moser flow: Divergence-based generative modeling on manifolds," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 669–17 680, 2021. 2

[26] G. Tiwari, D. Antic, J. E. Lenssen, N. Sarafianos, T. Tung, and G. Pons-Moll, "Pose-ndf: Modeling human pose manifolds with neural distance fields," in *European Conference on Computer Vision (ECCV)*, October 2022. 2

[27] S.-H. Yeh, D. Wang, W. Yan, and D. Song, "Detection of camera model inconsistency and the existence of optical image stabilization system," in *IEEE International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico*, August 2022. 4

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. 5

[29] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000. 5