LETTER

Single-Image Camera Calibration for Furniture Layout Using Natural-Marker-Based Augmented Reality

Kazumoto TANAKA[†a)], ***Member and*** **Yunchuan ZHANG**[††], ***Nonmember***

SUMMARY We propose an augmented-reality-based method for arranging furniture using natural markers extracted from the edges of the walls of rooms. The proposed method extracts natural markers and estimates the camera parameters from single images of rooms using deep neural networks. Experimental results show that in all the measurements, the superimposition error of the proposed method was lower than that of general marker-based methods that use practical-sized markers.

key words: *furniture layout, augmented reality, camera calibration, natural marker, deep neural network*

1. Introduction

Furniture layout is often considered when purchasing or renting living spaces. To enable comparisons of such spaces, it is convenient to consider virtual furniture arrangements, such as the advertised images of rooms on the Internet. Augmented reality (AR) offers solutions to such needs. This paper proposes an AR-based method for virtual furniture layout using single images of rooms.

The estimation of a matrix that transforms real-world coordinates into image system coordinates (i.e., camera calibration) is key to superimposing computer graphics models onto real-world images with accurate positions and orientations. Two types of methods are employed for matrix estimations in AR applications: marker-less and marker-based. Marker-less methods typically use visual simultaneous localization and mapping (VSLAM) [1] or visual-inertial simultaneous localization and mapping (VISLAM) [2]. However, these require multiple images and sensor data.

Marker-based methods generally estimate the transformation matrices using the geometric information of markers and camera geometry [3]. Their advantages include camera calibration with single images and markers representing the world coordinate system, such that the virtual object positions can be easily specified. However, placing such markers in individual rooms is difficult. Alternatively, indoor natural markers, such as exit signs, have been used as the AR markers [4]; however, such signs are not always available in all rooms. In contrast, this study proposes a natural-marker-based method. Specifically, five connected edges of the room's walls are used as the natural markers (Fig. 1, left), and camera calibrations are performed in the world coordinate system comprising three edges of these markers (Fig. 1, right).

Fig. 1 Left: Natural markers (black lines). Right: World coordinate system. The vertical line of the natural marker is the y-axis (blue), and the lines of intersections between the walls and floor are the x-axis (green) and z-axis (red). The axes are considered to form a right-handed system. When there are multiple candidates for the world coordinate system, the leftmost one in the image is selected in this study.

Several methods have been proposed for camera calibrations from single images, e.g. [5], using the edges of planes, such as walls, ceilings, and floors. Under the Manhattan world assumption [6], the camera orientations are estimated using three orthogonal directions obtained from the vanishing points of the edges. However, the camera positions were not determined in previous studies; therefore, the positions of virtual objects cannot be specified based on this coordinate system. In contrast, in this study, the camera pose (orientation and position) and view angle (corresponding to the focal length) are estimated from natural markers comprising the wall edges, and the world coordinate system-based position can thus be clearly shown to the user. To detect natural markers, we propose a deep neural network (DNN) named *Marker Detector* based on the image-to-image translation (pix2pix) framework [7].

Methods estimating the camera poses from single images using DNN have also been proposed [8]–[11]. PoseNet [8] outputs poses using the GoogLeNet architecture [12]. Hourglass Pose [9] utilizes ResNet34 [13] instead of GoogLeNet and improves the estimation accuracy via a decoding layer before the regression layer. MapNet [10] also uses ResNet34, but as a loss function, and both the per-image absolute orientation and relative orientation between image pair losses are used to improve the estimation accuracy. AtLoc [11] includes an attention module with a ResNet34-based network to achieve improved robustness against noise. These methods utilize camera relocalizations that learn the relationships between the image in a three-dimensional (3D) environment and camera pose to estimate

Manuscript received September 16, 2021.
Manuscript revised February 7, 2022.
Manuscript publicized March 9, 2022.

[†]The authors is with Faculty of Engineering, Kindai University, Higashihiroshima-shi, 739–2116 Japan.
[††]The authors is with Graduate School of Systems Engineering, Kindai University, Higashihiroshima-shi, 739–2116 Japan.
a) E-mail: kazumoto@hiro.kindai.ac.jp

DOI: 10.1587/transinf.2021EDL8086

the pose from any image of the given space.

In this study, the natural-marker image outputs from *Marker Detector* are used to obtain the camera parameters; however, as the 3D marker data are unknown, calibrating the camera in the same manner as the marker-based method is not possible. Therefore, we employ a DNN-based estimation method and propose a *Parameter Regressor* that outputs the camera pose and view angle using a modified ResNet34.

The main contributions of this study are as follows:

- An indoor AR method is proposed that has the advantages of the marker-based approach. It requires no special preparation and uses natural markers composed of the room edges.
- Using *Marker Detector* and *Parameter Regressor*, the camera pose and view angle are estimated from individual indoor images. To the best of our knowledge, there are no equivalent proposed methods for estimating the intrinsic and extrinsic parameters from single indoor images.

2. Method

The proposed camera calibration method has two main components: natural-marker detection and camera parameter estimation (Fig. 2: calibration phase). During marker detection, room images are converted to edge images by the Canny operator and then converted to natural-marker images using *Marker Detector*. The camera parameter estimation step then inputs the outputs from *Marker Detector* to *Parameter Regressor* and outputs the camera parameters. These proposed networks were trained from scratch.

2.1 Marker Detector

The edge images detected from input images have many edges, and it is necessary to extract only the natural-marker edges. Hence, *Marker Detector* directly outputs the natural-marker images using the pix2pix framework—a generative adversarial network comprising a generator that transforms input images to target images and a discriminator that checks if the output images are produced by the generator. This generator was used as *Marker Detector* in this study. The network architecture of *Marker Detector* is similar to the image transformation network in [14]. It consists of residual blocks [13] between downsampling and upsampling layers. Strided convolutions are used instead of pooling layers for downsampling and upsampling. Each convolution layer is followed by instance normalization and rectified linear unit (ReLU) layers, except for the last convolution layer—which is followed by a tanh output layer. The architecture is shown in Fig. 3. The discriminator is a PatchGAN [7] consisting of five convolution layers, each followed by instance normalization and ReLU; it outputs 16×9 patches.

The input and output images of *Marker Detector* are an edge image of a room and a natural-marker image with no noise, respectively. Consequently, the data for *Marker Detector* training comprise edge image pairs, each consisting of one image with partially and randomly missing natural marker and noise edges and a corresponding second image with only the natural marker (green part in Fig. 2).

The second images of the pairs are the ground truth for the other side of the pairs. They are generated by using OpenGL as follows:

- A rectangular cuboid wireframe is used as a room model. Regarding the scale, the metric used is room height, i.e., the length of the vertical edge of the natural marker is 1.0 (see Fig. 4). The length of the floor is 3.0, and there are 11 width types: 1.0, 1.2, ..., 3.0. Therefore, we use 11 room model types for the training.
  Step 1) Set the OpenGL space to black and the wireframe to white.
  Step 2) Divide the room space into a 3D grid with a mesh size of 0.2, randomly select one point from the inside of each voxel, and set it as an eye point (Fig. 4). Divide the wall on the x–y plane into a 2D grid with a mesh size of 0.2, randomly select one point from the inside of each cell, and set it as an aim point (Fig. 4). There are six view angle types: 40, 50, ..., 90 degrees. Add a perturbation randomly selected from the range of ±5 degrees to each angle when rendering.
  Step 3) For each room model, render the wireframe with all combinations of the above eye point, aim point,

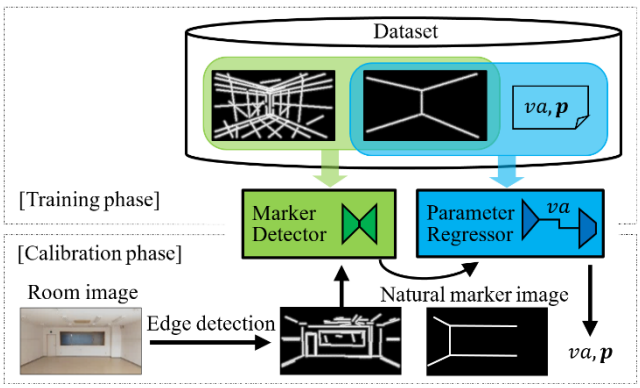

Fig. 2 Overview of the proposed method. *va* and ***p*** are the view angle and camera pose, respectively.

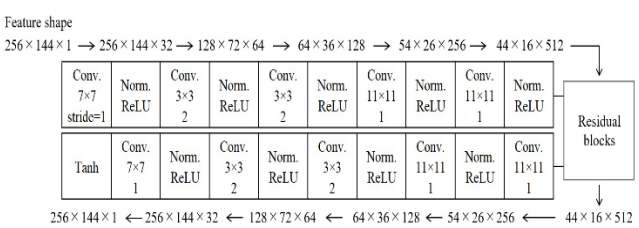

Fig. 3 Network architecture of Marker Detector. The residual blocks have nine blocks of convolution (kernel = 3×3, stride = 1), normalization, and ReLU.

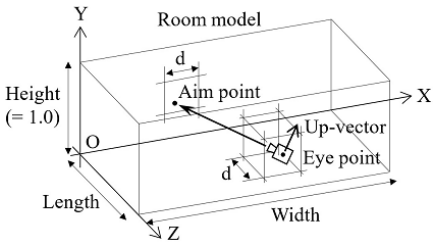

Fig. 4 World coordinate system, room model, and camera model. d = 0.2 (for training dataset) and 0.25 (for test dataset).

and view angle. Set the up-vector such that the rendered height edge is parallel to the y-axis of the image.
Step 4) Delete images that do not show the natural marker. (Blur the remaining images with a Gaussian filter (kernel size: 11 × 11) after Step 5 to make one of the image pairs. These images comprise the ground truth.)
Step 5) For all image duplicates obtained in Step 4, paint the randomly selected positions of the natural markers black to generate occlusions. Further, add random edges as noise to the images. Blur these images in the same manner to make the other side of the image pairs.

By using the above method, 252,000 image pairs of size 256 × 144 were generated as training data. Next, for test data, the room height and length were also 1.0 and 3.0 respectively, and there were four widths: 1.25, 1.75, 2.25, 2.75. The mesh size of the 3D and 2D grids was set to 0.25, and then 26,000 image pairs for test data were created by the random selection manner and the six types of view angles with perturbation.

In addition, we also utilized the ground truth images and the camera parameter used to generate the images for *Parameter Regressor* training. Thus, a dataset consisting of sets of the image pair and camera parameter was created as a shared dataset for *Marker Detector* and *Parameter Regressor* (the dataset in Fig. 2). The testing dataset was also constituted in the same manner.

The loss function for *Marker Detector* (generator) is as follows:

$$\|\mathbf{1} - \boldsymbol{D}(\boldsymbol{G}(\boldsymbol{x}))\|_2^2 + \lambda\|\boldsymbol{y} - \boldsymbol{G}(\boldsymbol{x})\|_2^2 \quad (1)$$

where $\|\cdot\|_2^2$, $\boldsymbol{G}$, $\boldsymbol{D}$, $\boldsymbol{x}$, $\boldsymbol{y}$, and λ denote the mean squared error function, the generator, discriminator, partially occluded natural-marker image with noise edges, ground truth image (image with only natural marker), and weighted parameter, respectively. The blur filtering in Steps 4 and 5 reduces the strictness of the pixel-wise loss [15]. The loss function for the discriminator is as follows:

$$\|\mathbf{1} - \boldsymbol{D}(\boldsymbol{y})\|_2^2 + \|\boldsymbol{D}(\boldsymbol{G}(\boldsymbol{x}))\|_2^2 \quad (2)$$

We trained *Marker Detector* for 150 epochs with no mini-batch. The Adam optimizer [16] was used with learning rate = 0.0002, and momentum parameters 1 and 2 set to 0.5 and 0.999, respectively. Figure 5 shows an example of the test results after the training.

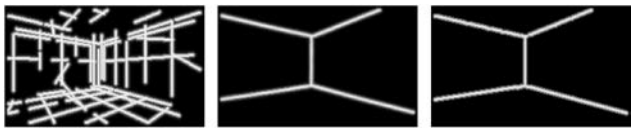

Fig. 5 Left: Partially occluded natural-marker image with noise edges. Middle: Image generated from the left image. Right: Ground truth.

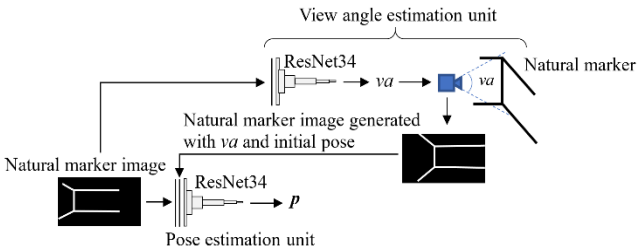

Fig. 6 Process flow of *Parameter Regressor*. The initial pose for generating the image from the view angle estimation unit was set as follows: eye point = (0.5, 0.5, 1.0), aim point = (0.5, 0.5, 0.0), and up-vector = (0.0, 1.0, 0.0).

2.2 Parameter Regressor

The ground truth images and camera parameters in the dataset are used for *Parameter Regressor* training (blue part in Fig. 2). The eye point is the camera position, and the combined line-of-sight vector from the eye point to the aim point and up-vector corresponds to the camera orientation. The number of estimation parameters is set to eight by setting the aim point on the z = 0 plane without loss of generality and setting the up-vector as a unit vector (element Y is limited to a positive value).

We employed ResNet34 as *Parameter Regressor*. However, training a network with the last layer of the fully connected layer modified to output the eight parameters yielded poor results. Therefore, from repeated attempts, we found that dividing *Parameter Regressor* into view angle and pose estimation units and supplying the view angle estimates to the pose estimation unit resulted in improvements. These two units utilize ResNet34 with one and seven outputs. The view angle estimation unit first estimates the view angle using the natural-marker image as input and generates an image with the view angle (estimated result) and initial pose. The estimated result is improved by concatenating the natural-marker image with the generated image and supplying it to the pose estimation network (Fig. 6). Surprisingly, good results were obtained even when the image with all the pixel densities set to the estimated view angle result was concatenated. Conversely, the results were not as good when the view angle was concatenated with the input to the fully connected layer.

In *Parameter Regressor* training, the view angle estimation unit and the pose estimation unit were trained for 700 epochs and 600 epochs with eight mini-batch sizes, re-

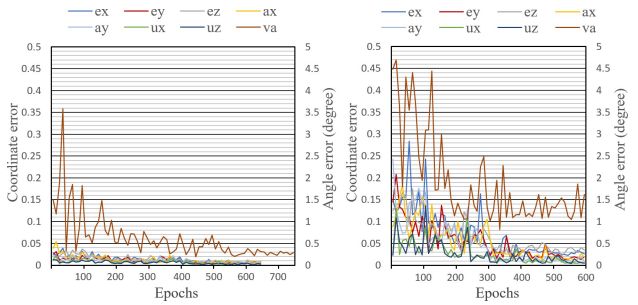

Fig. 7 Test errors for parameter estimation based on the proposed method (left) and baseline (right). (ex, ey, ez), (ax, ay), and (ux, uz) denote eye point, aim point, and up-vector, respectively. The metric used is room height; thus, if the height is 2300 mm, a coordinate error of 0.01 equates to an actual error of 23 mm.

spectively. Subsequently, each unit was additionally trained for another 50 epochs on natural-marker images that *Marker Detector* generated from the partially occluded natural-marker image with noise edges in the dataset. In the additional training, the trained *Marker Detector* was used, and the parameters of *Marker Detector* were fixed.

For all the training of the two units, the Adam optimizer with the same parameters as those used for *Marker Detector* was used. The loss functions for the view angle estimation unit and the pose estimation unit are the mean squared error functions between estimated values and ground truth values. The test errors of the parameter estimation are shown in Fig. 7 (left). The additional training had little effect on the estimation accuracy.

To demonstrate the effectiveness of the proposed network architecture consisting of the two estimation units, the modified ResNet34, which outputs eight parameters (including view angle) from the natural marker, was set as a baseline. Figure 7 (right) shows the results of 600 epochs (a weighted parameter for the view angle was employed for the loss function). The results indicate that the convergence accuracy of the baseline is worse than that of the proposed method. For the camera parameter estimation based on reprojection error minimization, there is some concern regarding the simultaneous estimation of intrinsic parameters and extrinsic parameters [17]. That is, the accuracy of the intrinsic parameters and that of the extrinsic parameters affect each other, such that the accuracy is lower than when estimating separately. Although our proposed approach is different, both parameters are not completely independent as they (except up-vector) relate to the size of the natural marker image. Therefore, the proposed method of dividing into two units is effective.

3. Experiments

Virtual furniture were superimposed on the room images for visual evaluation of the proposed method. Then, the superimposed accuracy between the proposed and marker-based methods was compared for objective evaluations.

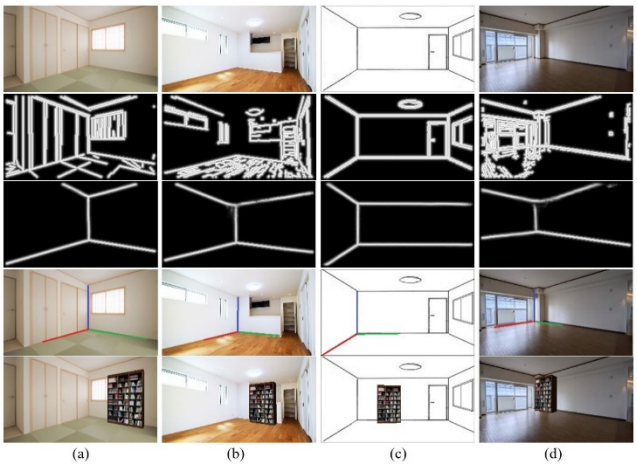

Fig. 8 Top to bottom: original image, edge detection by Canny detector, output image of *Marker Detector*, world coordinate system (axis length = 1.0) reprojected using estimated results of *Parameter Regressor*, and superimposed results for virtual bookshelf.

3.1 Superimposition on Room Images

Using sample images of vacant rooms from the Internet (https://www.photo-ac.com/) and illustration images, we conducted an experiment in which a virtual bookshelf was superimposed at a specific position. Some example results are shown in Fig. 8. In each case, the upper left vertex of the bottom of the bookshelf was specified by OpenGL rendering for placement at the coordinates $(0.5, 0.0, 0.0)$ of the natural-marker coordinate system. The results show that if the edges of the original image corresponding to the natural markers are not hidden (top of Figs. 8 (a), 8 (b), and 8 (c)), then superimposition can be performed accurately even when edge detection is incomplete (second from top in Fig. 8 (b)). However, if the edges of the corners are hidden by structures such as pillars, then the natural-marker estimation fails (Fig. 8 (d)). Hence, even illustrations can be superimposed without any problem when drawn in perspective view (Fig. 8 (c)).

3.2 Comparison of Accuracy with Marker-Based Method

For the evaluation of the proposed method, we compared the accuracy of reprojection using the OpenCV ArUco module [18] as a general marker-based method. The experimental settings and method were as follows. Assuming the normalized dimensions of a standard living room to be 1.5 (length) × 1.5 (width) × 1.0 (height), we used a space of 435 mm ×435 mm ×290 mm as a model for the living room in the experiment. When an image of the living room is taken from the wall with a camera having a view angle of 80 degrees, a space with depth (z-axis direction: see Fig. 4) approximately 1/3 can be covered completely. Consequently, six equally spaced reference points ranging from $(0.50, 0.00, 0.05)$ to $(1.50, 0.00, 0.55)$ on the

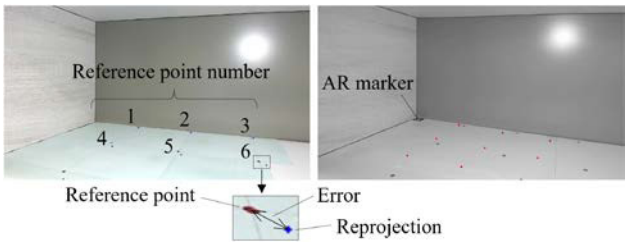

Fig. 9 Room space model and six reference points. Upper left: proposed method; Upper right: ArUco marker-based method.

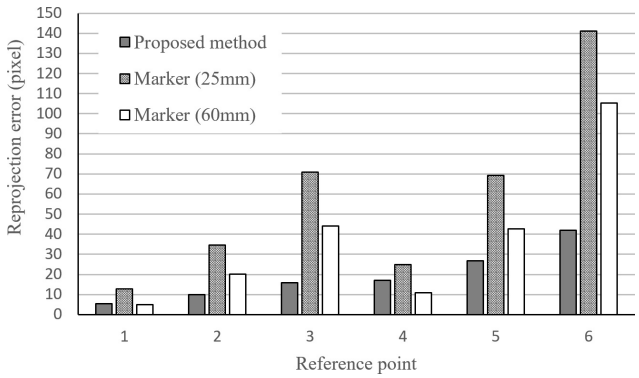

Fig. 10 Reprojection errors (pixels) for the 1024×576 image size. The room space model was taken with a Full-HD camera, and images were resized to 256×144 and 1024×576 for the proposed method and ArUco marker-based method, respectively. For the proposed method, the reprojected coordinates were converted for a 1024×576 image.

floor of the model (actual size: (145 mm, 0 mm, 14.5 mm)–(435 mm, 0 mm, 159.5 mm)) were used for reprojection (see Fig. 9).

Generally, a compact AR marker is better, but if it is overly small, there will be a problem with the recognition rate of the marker. As the minimum size that can be stably recognized in this experimental environment was 25 mm (when using a full high-definition camera), ArUco marker of this size was used. The 25 mm marker corresponds to a marker of approximately 198 mm in a room with a height of 2300 mm. This is convenient for printing on a letter-size sheet of paper and also for carrying.

We selected 20 camera positions and orientations such that the six reference points and the edge in the height direction of the room model could be captured by a full high-definition camera with a view angle of 80 degrees. Further, we took images of the reference points with the 20 camera poses. Using these images, six points were reprojected onto each image using the camera parameters estimated by the proposed method and the marker method (Fig. 9). Figure 10 shows the average reprojection error at each point. The results show that the proposed method is highly accurate for all reference points and is more useful than the practically sized marker methods.

Furthermore, ignoring practicality, a similar experiment was conducted with a 60 mm marker (that corresponds to 476 mm at 2300 mm height). This result (Fig. 10) also shows that the accuracy of the proposed method is higher except for points 1 and 4 near the origin.

4. Limitations and Conclusions

This paper proposed an AR-based method for arranging furniture using natural markers without the need for special preparation. The proposed method extracts natural markers and estimates the camera parameters from single images of rooms using DNNs. The method requires images containing edges of the room to derive the natural marker; additionally, the walls must be mutually orthogonal. Although the proposed method has these limitations, it can still be used to specify the positions for placing virtual objects based on the coordinate system of the natural marker; further, this method is more accurate than other marker-based methods that use practical-sized markers. Future work will mainly tackle the estimation failure caused by corner edge occlusion.

Acknowledgements

This work was partially supported by JSPS KAKENHI Grant Number 21K12166.

References

[1] Q.H. Gao, T.R. Wan, W. Tang, and L. Chen, "A stable and accurate marker-less augmented reality registration method," Proc. Int'l Conf. on Cyberworlds, Chester, pp.41–47, 2017.
[2] S. Tomažič and I. Škrjanc, "An automated indoor localization system for online bluetooth signal strength modeling using visual-inertial SLAM," Sensors, vol.21, no.8, 2857, 2021.
[3] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," Proc. 2nd IWAR, San Francisco, pp.85–94, 1999.
[4] C. Koch, M. Neges, M. König, and M. Abramovici, "Natural markers for augmented reality-based indoor navigation and facility maintenance," Automation in Construction, vol.48, pp.18–30, 2014.
[5] W. Elloumi, S. Treuillet, and R. Leconge, "Real-time camera orientation estimation based on vanishing point tracking under Manhattan world assumption," Journal of Real-Time Image Processing, vol.13, pp.669–684, 2017.
[6] J.M. Coughlan and A.L. Yuille, "The Manhattan world assumption: Regularities in scene statistics which enable Bayesian inference," Proc. Adv. Neural Inf. Process. Syst., Denver, pp.845–851, 2000.
[7] P. Isola, J.-Y. Zhu, T. Zhou, and A.A. Efros, "Image-to-image translation with conditional adversarial networks," Proc. CVPR2017, Honolulu, pp.5967–5976, 2017.
[8] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-dof camera relocalization," Proc. ICCV2015, Santiago, pp.2938–2946, 2015.
[9] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, "Image-based localization using hourglass networks," Proc. ICCV2017, Venice, pp.870–877, 2017.
[10] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-aware learning of maps for camera localization," Proc. CVPR2018, Salt Lake, pp.2616–2625, 2018.
[11] B. Wang, C. Chen, C.X. Lu, P. Zhao, N. Trigoni, and A. Markham, "AtLoc: Attention guided camera localization," Proc. AAAI2020, New York, pp.10393–10401, 2020.
[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," Proc. CVPR2015, Boston, pp.1–9, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proc. CVPR2016, Las Vegas, pp.770–778, 2016.
[14] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," Proc. ECCV2016, Amsterdam, pp.694–711, 2016.
[15] A. Wang, M. Ren, and R.S. Zemel, "SketchEmbedNet: Learning novel concepts by imitating drawings," Proc. ICML2021, (Virtual), 139:10870-10881, 2021.
[16] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Proc. ICLR2015, San Diego, arXiv:1412.6980, 2015.
[17] R. Sagawa and Y. Yagi, "Accurate calibration of intrinsic camera parameters by observing parallel light pairs," Proc. ICRA2008, Pasadena, pp.1390–1397, 2008.
[18] https://docs.opencv.org/4.x/d9/d6d/tutorial_table_of_content_aruco.html