Computational Visual Media
DOI 10.1007/s41095-016-0067-z
Vol. 3, No. 2, June 2017, 189–198

Research Article

Robust camera pose estimation by viewpoint classification using deep learning

Yoshikatsu Nakajima[1] (✉), **Hideo Saito**[1]

Abstract Camera pose estimation with respect to target scenes is an important technology for superimposing virtual information in augmented reality (AR). However, it is difficult to estimate the camera pose for all possible view angles because feature descriptors such as SIFT are not completely invariant from every perspective. We propose a novel method of robust camera pose estimation using multiple feature descriptor databases generated for each partitioned viewpoint, in which the feature descriptor of each keypoint is almost invariant. Our method estimates the viewpoint class for each input image using deep learning based on a set of training images prepared for each viewpoint class. We give two ways to prepare these images for deep learning and generating databases. In the first method, images are generated using a projection matrix to ensure robust learning in a range of environments with changing backgrounds. The second method uses real images to learn a given environment around a planar pattern. Our evaluation results confirm that our approach increases the number of correct matches and the accuracy of camera pose estimation compared to the conventional method.

Keywords pose estimation; augmented reality (AR); deep learning; convolutional neural network

1 Introduction

Since augmented reality (AR) toolkit [1] introduced the superimposition of virtual information onto planar patterns in images by real-time estimation of camera pose, technologies for markerless camera-tracking technology have become mainstream [2, 3]. Markerless tracking needs to find a point of correspondence between the input image and the planar pattern for any camera pose.

Lowe's SIFT [4] is one of the most famous algorithms in computer vision for detecting keypoints and describing local features in images. SIFT detects keypoints using differences of Gaussians to approximate a Laplacian of Gaussian filter and describes them using a 128-dimensional feature vector. Then, keypoint correspondences are obtained using Euclidean distances between feature vectors. Although SIFT is robust in the face of scaling and rotation [5], when the input image is distorted due to projection distortion of the planar pattern, we cannot find keypoint correspondences. *Randomised trees* (RT) [6] improve the problem by training a variety of descriptors for each keypoint using affine transformations, and generating a tree structure [7] based on the resulting brightness values, for real-time recognition of keypoint identity. *Viewpoint generative learning* (VGL), developed by Yoshida et al. [8], extends this idea to train various descriptors for every keypoint by generating images as if they were taken from various viewpoints using a projection transformation, and generating a database of keypoints and features from the images.

However, methods based on training feature descriptors of keypoints, such as RT and VGL, trade robustness of the various descriptors against computation time when searching for matched keypoints. For example, VGL compresses the database of training descriptors using k-means clustering [9] for fast search, but this sometimes results in wrong keypoint matching, especially when the camera angle is shallow. Because feature descriptors of keypoints change significantly at a

[1] Department of Science and Technology, Keio University, Japan. E-mail: Y. Nakajima, nakajima@hvrl.ics.keio.ac.jp (✉); H. Saito, saito@hvrl.ics.keio.ac.jp.

Manuscript received: 2016-07-25; accepted: 2016-11-13

shallow angle, weak compression of the database is required to allow such shallow camera angles, but this increases the computation for keypoint search.

In this paper, we propose a novel method for camera pose estimation based on two-stage keypoint matching to solve the trade-off problem. The first stage is viewpoint classification using a *convolutional neural network* (CNN) [10–12], so that the feature descriptors of every keypoint are similar from the classified viewpoints. The second stage is camera pose estimation based on accurate keypoint matching, which is achieved in the classified viewpoint using a nearest neighbor (NN) search for the descriptor. To achieve this two-stage camera pose estimation, in pre-processing, our method generates the uncompressed descriptor databases of a planar pattern for each partitioned viewpoint, including a shallow angle, and trains a CNN to classify the viewpoint of the input image.

A CNN can perform stable classification against variations of a property for the same class by learning from a large amount of data with variations for each class. For instance, object recognition which is stable under viewpoint changes can be performed by learning from many images taken from various viewpoints for each object class [13]. This stable performance against viewpoint change is not achieved by just the structure of the CNN, but through the capability of a CNN to learn from variable data for each class. For example, Agrawal et al. [14] applied a CNN to estimate egomotion by constructing a network model with two inputs comprising two images whose viewpoints slightly differ. In this paper, we apply a CNN to a viewpoint classification for a single object.

Additional reasons for using a CNN for viewpoint classification are as follows. Firstly, a CNN is robust to occlusion. This is very important, as it widens the range of applications. Secondly, computation time is unchanged as the number of viewpoint classes increases, enabling us to easily analyze the trade-off relationship between accuracy and size.

We introduce two methods for generating a database and preparing the images for deep learning, under the assumption that those methods might be used in different ways. The first one is robust for a range of environments and is used to initialize camera pose estimation, etc. The second method learns the entire environment around the planar pattern and is used in a learned environment.

The NN search in the second stage is not time-consuming, because little variety is necessary in the descriptors in the classified viewpoint in the first stage. The camera pose of the input image is computed based on correspondences between matched keypoints.

2 Method

Figure 1 shows the flow of our proposed method, which consists of three parts. The first part generates databases of features for every viewpoint class, which are partitioned into viewing angles from the entire viewing angle range $(-90^\circ < \theta < 90^\circ, -180^\circ < \phi < 180^\circ)$ with respect to a target planar pattern, as shown in Fig. 2. The second part trains the CNN to classify the viewpoint of the input image. The last part estimates the camera pose of the input image. We now explain each part in detail.

In particular, during database generation (Section 2.1) and CNN training (Section 2.2), we use two methods to prepare images for database generation and deep learning by the CNN, with the assumption that these methods are to be used in different ways.

The first method uses only one image of the planar pattern and generates many images by use

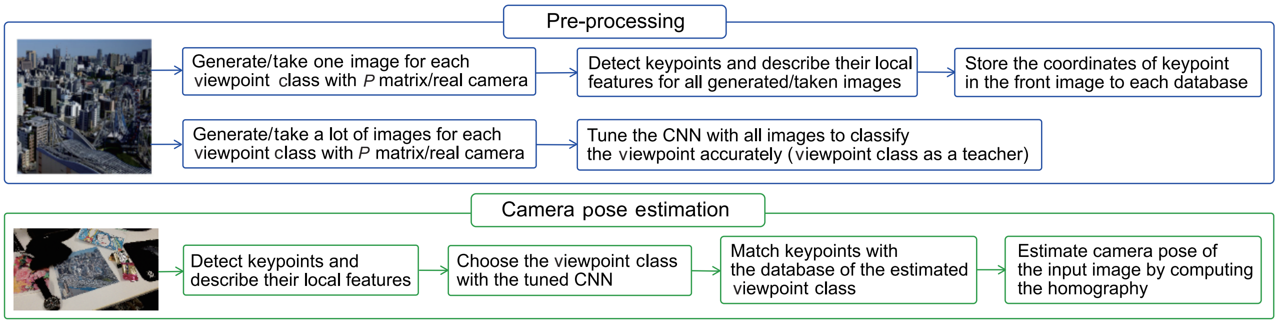

Pre-processing
Generate/take one image for each viewpoint class with P matrix/real camera
Detect keypoints and describe their local features for all generated/taken images
Store the coordinates of keypoint in the front image to each database
Generate/take a lot of images for each viewpoint class with P matrix/real camera
Tune the CNN with all images to classify the viewpoint accurately (viewpoint class as a teacher)
Camera pose estimation
Detect keypoints and describe their local features
Choose the viewpoint class with the tuned CNN
Match keypoints with the database of the estimated viewpoint class
Estimate camera pose of the input image by computing the homography

Fig. 1 Flow of proposed method. Top: database generation, middle: deep learning by the CNN, bottom: camera pose estimation.

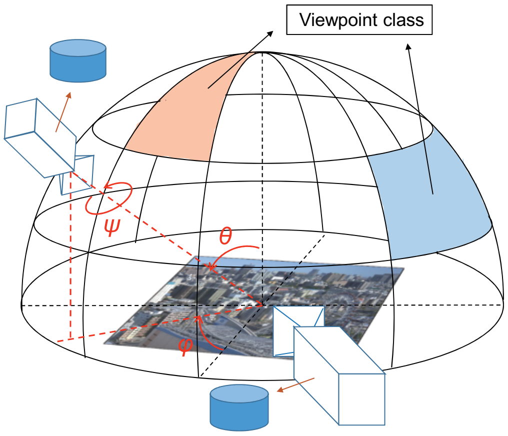


Fig. 2 Generating databases: viewpoint class, virtual camera, and angle definitions.

of projection matrices (P matrices). This reduces the learning cost because it only uses a single image and P matrices. Moreover, this enables the CNN to be robust with changes in background of the input image, because we can vary the backgrounds of the images generated for deep learning. Viewpoint class estimation will not be extremely accurate; however, because the CNN only uses the appearance of the planar pattern in the input image, so this first method is not suitable for movies. On the other hand, it can manage shallow angles better than the conventional method, so it is useful for the initialization of the camera pose, etc. From now on, we call this method *learning based on generated viewpoints.*

The second method uses real images by fixing the planar pattern within the environment and taking pictures with a camera. The CNN can learn not only about the appearance of the planar pattern but also the environment around it, including the background, the lighting, and so on. Therefore, the viewpoint class of the input image can be estimated with almost perfect precision, so this method is suitable for movies. However, the CNN can be only used in the environment in which the planar pattern is fixed when images for deep learning are taken. In contrast to the first method, we call this method *learning based on example viewpoints.*

2.1 Database generation

In this part, we generate one feature database per viewpoint class. Each database is generated from one image because features sampled from a certain viewpoint are almost identical in the viewpoint class, so one image is enough. As mentioned in the introduction, we use two methods for preparing images for database generation. Firstly, we will explain the method using one image and a P matrix, which is robust in various environments. Secondly, we will explain the method using real images taken by a camera, which is more robust in the particular environment in which the pre-processing is performed. This flow is shown in the upper part of Fig. 1.

2.1.1 Learning based on generated viewpoints

Firstly, we partition the entire range of viewing angles of the camera's viewpoint with respect to a target pattern. We call each partitioned viewpoint a viewpoint class (see Fig. 2). Secondly, we compute the projection matrices that transform the frontal image to images which appear to have been taken from the center of each viewpoint class, using Eq. (1). From now on, we will denote the number of viewpoint classes by N, the viewpoint classes by V_i ($i = 1, \ldots, N$), and the projection matrix for each viewpoint class V_i by $\boldsymbol{P}_i$. In Eq. (1), let the intrinsic parameters of the virtual camera, the rotation matrix for viewpoint class V_i, and the translation vector, be $\boldsymbol{A}$, $\boldsymbol{R}_i$, and $\boldsymbol{t}$, respectively. The matrix $\boldsymbol{R}_i$ is given by Eq. (2), using θ, ϕ, and ψ defined as in Fig. 2.

$$\boldsymbol{P}_i = \boldsymbol{A}(\boldsymbol{R}_i|\boldsymbol{t}) \tag{1}$$

$$\boldsymbol{R}_i = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\phi & \theta & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \tag{2}$$

Using the projection matrix $\boldsymbol{P}_i$, we obtain an image I_i for each viewpoint class V_i. Next, we detect keypoints and describe their local features for each image I_i, using the appropriate algorithm. We denote the number of detected keypoints by M_i, each keypoint by p_{ij}, and each feature by d_{ij} ($j = 1, \ldots, M_i$). Then we compute a homography matrix $\boldsymbol{H}_i$ that transforms the image I_i, which represents the viewpoint class V_i, to the frontal image. We also generate the database in which the described features d_{ij} and their coordinates p'_{ij} in the frontal image are stored. The coordinates p'_{ij} are found by transforming the coordinates of each detected keypoint p_{ij} to the frontal image using the equation $p'_{ij} = \boldsymbol{H}_i p_{ij}$. By performing this process on all images that represent each viewpoint class, we obtain one uncompressed descriptor database per viewpoint class.

2.1.2 *Learning based on example viewpoints*

For this method, we first use the camera to take multiple viewpoints of the planar pattern that is fixed in the environment. Next, for each image I_i, we compute a homography matrix $\boldsymbol{H}_i$ that transforms the image I_i to the frontal image. In this computation, we use four points whose coordinates in the frontal image are easily determined, like corners. Equation (3) can be used to compute the homography matrix $\boldsymbol{H}_i$:

$$\tilde{\boldsymbol{x}}' \sim \boldsymbol{H}\tilde{\boldsymbol{x}} \tag{3}$$

Here, we denote the coordinates in the frontal image of the planar pattern by $\tilde{\boldsymbol{x}}' \sim (x', y', 1)^{\mathrm{T}}$ and the coordinates in the taken image as $\tilde{\boldsymbol{x}} \sim (x, y, 1)^{\mathrm{T}}$. Then, we detect keypoints and describe their local features in the image I_i using the appropriate algorithm. We denote the number of detected keypoints by M_i, each keypoint by p_{ij}, and each feature by d_{ij} $(j = 1, \ldots, M_i)$. The keypoint p_{ij} can be projected into p'_{ij}, which represents the coordinates in the frontal image, using $p'_{ij} = \boldsymbol{H}_i p_{ij}$. Finally, we generate the database for each image; multiple sets of p'_{ij} and d_{ij} are stored. By judging whether the coordinates of p'_{ij} are on the planar pattern or not, we can eliminate features belonging to the environment when we store features belonging to the planar pattern in the database.

2.2 Deep learning by the CNN

We train a CNN for the purpose of classifying the viewpoint of the input image. A CNN is a deep neural network mainly used for object recognition. We apply a CNN to viewpoint classification of a single planar pattern. In this step, we only use images, and do not use features for deep learning, because we employ a CNN that only receives images as input. As we do with database generation, we will explain the two methods of preparing images for deep learning. However, the deep learning processing explained below should use the same method as that used for the database generation step. This process is illustrated in the middle row of Fig. 1.

2.2.1 *Learning based on generated viewpoints*

Firstly, we generate multiple images for each viewpoint class V_i using Eq. (1). Then we randomly change the background of every image and the position and scale of the planar pattern. By using these images for deep learning, the weight of the background part is reduced and the CNN can classify the viewpoint robustly. Here, we employ a softmax function as the activation function of the output layer and make its number of units coincide with the number of viewpoint classes—this is the CNN design recommended for classification problems. Finally, we perform deep learning by teaching the CNN the correct viewpoint class for each generated image using the techniques of back-propagation [15], pre-training [16], and drop-out [17]. In general, it is a problem for deep learning to prepare images for training, but our method uses images synthesized from a single planar pattern, enabling us to reduce the learning cost.

2.2.2 *Learning based on example viewpoints*

For each image I_i, we take multiple images of the planar pattern for deep learning from the same viewpoint as that used for image I_i. We then change the scale and the rotation to help ensure that the CNN is robust. Deep learning is performed as in Section 2.2.1, i.e., we employ a softmax function in the output layer, we make its number of units coincide with the number of the viewpoint classes, and we teach the CNN the correct viewpoint class for every image.

2.3 Camera pose estimation

In this section, we explain the details of camera pose estimation given the input image. This process is shown at the bottom of Fig. 1. We detect keypoints and describe their local features in the image using the same algorithm as that used to generate the databases. Next, we input the image to the CNN, which has been tuned by deep learning. Because the activation function of the output layer is a softmax function, the percentage informs us which viewpoint class the image belongs to (see Fig. 3). We select the viewpoint class with the highest percentage and compare keypoints in the database for that viewpoint class with keypoints in the input image in terms of the Euclidean distance of their feature descriptors. Then we search for the nearest keypoint and the next nearest as suggested by Mikolajczy et al. [18], so that we can use the ratio to reduce mismatches between keypoints. Only when the Euclidean distance to the nearest keypoint is sufficiently smaller than the Euclidean distance to the second one, there is a match. Thus, D_{A} and D_{B} are matched only when

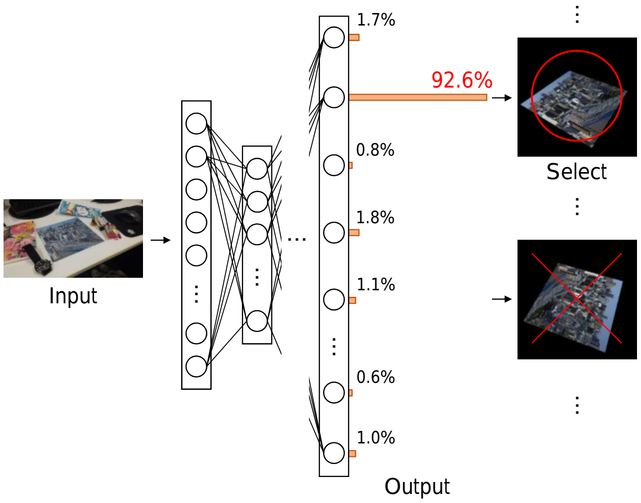


Fig. 3 Viewpoint class estimation using CNN.

Eq. (4) is satisfied:

$$\frac{\|D_{\mathrm{A}}-D_{\mathrm{B}}\|}{\|D_{\mathrm{A}}-D_{\mathrm{C}}\|}<t \tag{4}$$

Here, D_{A}, D_{B}, and D_{C} represent the feature descriptor of the input image, the feature descriptor of the nearest keypoint in the database, and the feature descriptor of the second nearest, respectively. If we set the threshold t large, the number of matches increases as well as the number of mismatches; conversely, if we set the threshold t small, the number of matches reduces as well as the number of mismatches.

By matching keypoints between the database and the input image, we can obtain corresponding points in the input image and the frontal image, as feature descriptors and their coordinates in the frontal image are stored in each database. After mismatches are reduced by RANSAC [19], we estimate the camera pose of the input image by computing the homography that transforms the frontal image to the input image using the coordinates of those correspondinge points.

3 Experimental evaluation

In this section, we demonstrate the validity of our method through experiments. In Section 2, we introduce two methods of preparing images for database generation and deep learning. Because those methods have different uses, we evaluate them with different datasets. We use VGL [8] as a basis for comparison. Conventional methods of camera pose estimation with CNN are typified by PoseNet, as described by Kendall et al.; however, such a method does not use SIFT-like point-based features, while VGL does use point-based matching. Furthermore, VGL is more robust than other conventional methods that use point-based matching like ASIFT [20] and *random ferns* [21]. Thus, we compare our method to VGL.

3.1 Experimental setup

The evaluation environment was as follows. CPU: Intel Core i7-4770K 3.5 GHz, GPU: GeForce GTX760, and RAM: 16 GB. The definition of viewpoint class and the datasets are different for the two methods, and will be explained separately. The deep learning framework used in this evaluation experiment was Chainer [22].

3.1.1 Learning based on generated viewpoints

For this method, we defined the viewpoint class V_i by splitting the viewpoints for observing the planar pattern as shown in Table 1. As features change more at a shallow angle, we subdivided the viewpoint more as angle θ increased. Thus, the number of viewpoint classes was $4+8+12+12=36$ in this experiment.

As for ψ, due to use of rotation invariant features including SIFT, we obtained many keypoint matches between the input image and the database for every values of camera pose angle ψ for the input image.

Next, we generated images I_i to represent each viewpoint class V_i using Eq. (1), using angles θ and ϕ at the center of each viewpoint class V_i. We used SIFT to detect keypoints p_{ij} and describe their local features d_{ij}. We used *network-in-network* (NIN) [23] to constitute the CNN. NIN is useful for reducing classification time by reducing the number of parameters while maintaining high accuracy. To tune the parameters of the CNN by deep learning, we generated about three thousand images for each viewpoint class V_i. The background images were prepared by capturing each frame from a movie taken indoors. Furthermore, we randomly changed the radius of the sphere (see Fig. 2) and the angle ψ when we generated the images for deep learning. Doing so allows estimation of the viewpoint class with the trained CNN even if the camera distance and the

Table 1 Viewpoint class definition

Range of θ	0°–19°	20°–39°	40°–59°	60°–79°
Partitions of ϕ	4	8	12	12

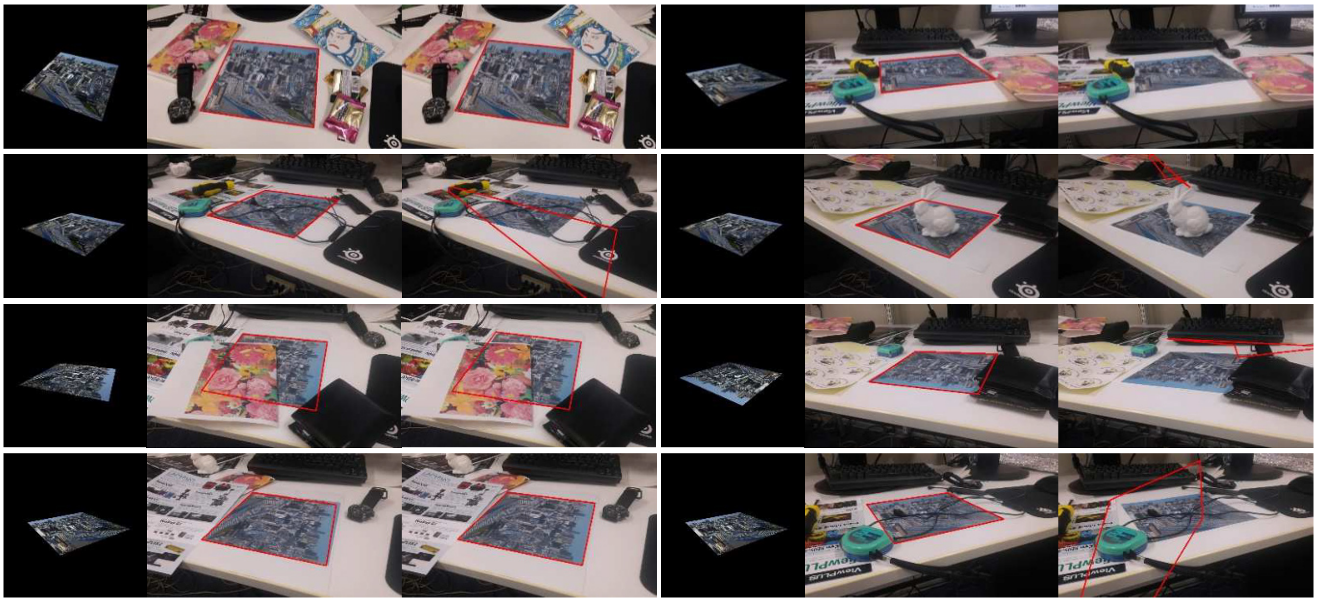


Fig. 4 Viewpoint class and camera pose estimation results using our method and VGL [8], using evaluation images. Left: estimated viewpoint class, center: our method, right: VGL [8].

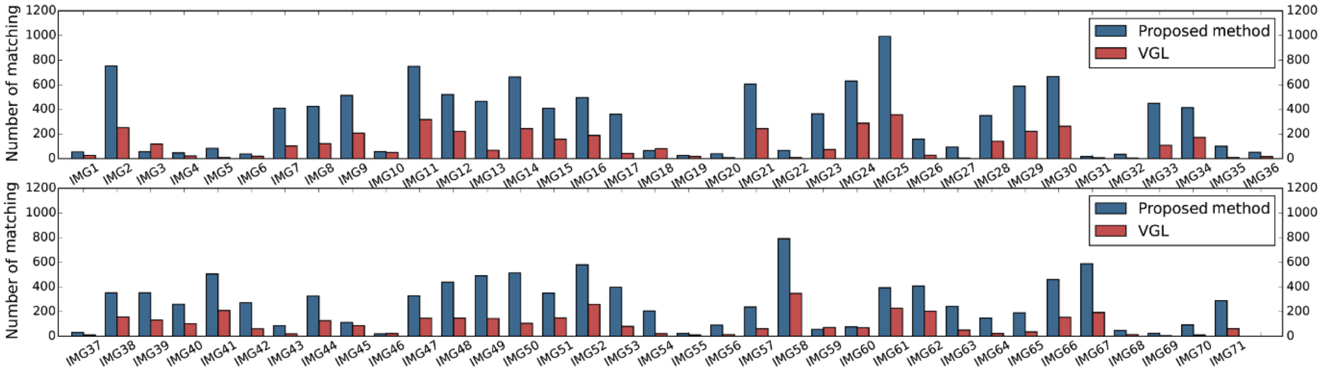


Fig. 5 Number of keypoint matches.

camera orientation with respect to the input image change.

We prepared 71 images of the planar pattern, including ones taken from a shallow angle and ones in which the planar pattern was occluded. Using those images, we compared the accuracy of camera pose estimation, the number of correct matches, and the processing time with the corresponding values for VGL. For VGL, we generated a database using the same images as in our method and set the number of clusters to five and the number of stable keypoints to 2000.

3.1.2 Learning based on example viewpoints

In this method, we took 22 images I_i ($N = 22$) of a planar pattern from multiple viewpoints after fixing the planar pattern onto a desk. These images define the viewpoint class V_i (see Fig. 2), so there were 22 viewpoint classes in this experiment. Using those images I_i, we generated 22 feature databases containing the coordinates p'_{ij} of all detected keypoints and their local features d_{ij} . We employed SIFT as a keypoint detector and a feature descriptor, and used the coordinates of four corners to compute $\boldsymbol{H}_i$ used to transform coordinates p_{ij} to coordinates p'_{ij}. We again employed NIN as the network model for the CNN. Next, we generated about 600 images for each viewpoint class V_i by clipping every frame of movies that we took from around the viewpoint of each of the 22 images I_i. By teaching the correct viewpoint class for every prepared image to the CNN using deep learning, the CNN became able to estimate the viewpoint class for each input image. Again in this method, we randomly changed the camera distance and the angle

ψ when we prepared the images for deep learning to make the CNN robust to changes in scale and rotation.

For the evaluation experiment, we prepared a movie of the fixed planar pattern, including frames taken from a shallow angle, in the same environment as the one used for database generation and image preparation for deep learning.

In this experiment, we evaluated the estimated camera pose from the re-projection error of the corners of the planar pattern. Denoting the coordinates of the corners observed in the test image by P_k, and the coordinates of the corners re-projected using the estimated homography $\boldsymbol{H}$ by Q_k, the re-projection error E is given by the following equation:

$$E = \sqrt{\frac{1}{4}\sum_{k=1}^{4} \|P_k - Q_k\|^2} \tag{5}$$

E represents the average Euclidean distance of the four corners between the ground-truth coordinates and the estimated coordinates. Minimising E gives the camera pose estimation.

To compare VGL with this method, we generated a database with the same 22 images used for our method and set the number of clusters to five and the number of stable keypoints to 2000.

3.2 Results

We now describe the results of the experimental evaluation of each method.

3.2.1 Learning based on generated viewpoints

Figure 4 shows the results of viewpoint class estimation by the CNN, and camera pose estimation using our method and VGL. The left image indicates the viewpoint class estimated by the CNN for the input image, the center image shows the result of camera pose estimation by our method, and the right image is the result from VGL. We visualize camera pose estimation by re-projecting coordinates of the four corners of the frontal image using the computed homography and connecting them with red lines. Images without red lines indicate lacked sufficient matches to compute the homography. Figure 5 gives the number of keypoint matches used to compute the homography for each of the 71 images.

As Fig. 4 shows, our method estimated camera pose more robustly for shallow angles than the conventional method. Figure 5 shows that the number of matches was higher for our method than for the conventional method, for almost all images. Because our method matches keypoints between the input image and a database that was generated using an image similar to the input image, matching was more accurate with our method. Although the planar pattern is occluded in some images in Fig. 4, our method estimated the viewpoint class and camera pose accurately. Because deep CNNs give robust results in the presence of occlusion [13], and the uncompressed descriptor databases of the planar pattern are generated for each viewpoint class, our method was robust to occlusion.

Regarding the accuracy of viewpoint classification, 7 of 71 images were incorrectly classified. However, 4 of these 7 images were successfully classified into the adjacent viewpoint class, so that keypoint matching works well enough, and the camera pose is estimated reasonably precisely. Features in the adjacent viewpoint class are similar to features in the correct one since the database for the next viewpoint class is generated from an image taken from a viewpoint next to the correct viewpoint. This allows the second step of accurate localization to still have a chance to correct the errors, making the algorithm robust. In contrast, 3 of 7 images were classified into a completely different viewpoint class, so camera pose estimation failed. Overall, viewpoint class estimation was about 90% accurate because the CNN only uses the appearance of the planar pattern in the input image. Therefore, this method is not suitable for movies. On the other hand, it copes with shallow angles (see Fig. 4), so it is useful for initialization of the camera pose and similar tasks. Furthermore, when using our method in applications, we can easily combine it with a conventional tracking method. By doing that, we can estimate camera pose continuously while coping with shallow angles.

Next, we consider processing time. Table 2 shows the average processing time for our method and VGL for all images, for each stage of camera pose estimation, and in total.

The overhead for viewpoint class estimation in our method is small. Detecting keypoints and describing feature descriptors using SIFT account for the most of the processing time. We could easily apply our method to a binary algorithm

Table 2 Average time spent on each processing stage

(Unit: ms)

	Our method	VGL
SIFT	201	214
Viewpoint class estimation	9	—
Matching	35	28
Homography computation	10	2
Total	255	244

like AKAZE [24] because it generates uncompressed descriptor databases. Thus, we could reduce the processing time spent on detecting keypoints and describing features in our method.

3.2.2 Learning based on example viewpoints

Figure 6 shows some results of camera pose estimation using our method and VGL, for a movie that we prepared for this evaluation. The camera pose was estimated using the method described in Section 3.2.1. Figure 7 shows the re-projection error computed with Eq. (5) for each frame of the movie. The ground-truth coordinates of the corners were detected manually. Figure 8 shows the number of keypoint matches between the input image and the estimated ones used for computation of the homography.

As shown in Fig. 6, this method also estimated camera pose for shallow angles more robustly than the conventional method. In Fig. 8, the number of matches fluctuates because the database used for keypoint matching was changed by the CNN every few frames. As shown by Figs. 6–8, the accuracy of camera pose estimation using VGL decreased for shallow angles and the features changed drastically, because VGL compresses features using k-means for fast computation. On the other hand, our method estimates the camera pose more robustly because the database that contains all features sampled from images similar to the input image is appropriately selected by the CNN.

With this method, viewpoint class estimation accuracy is almost 100% (see Fig. 7), because the CNN can learn not only the appearance of the planar

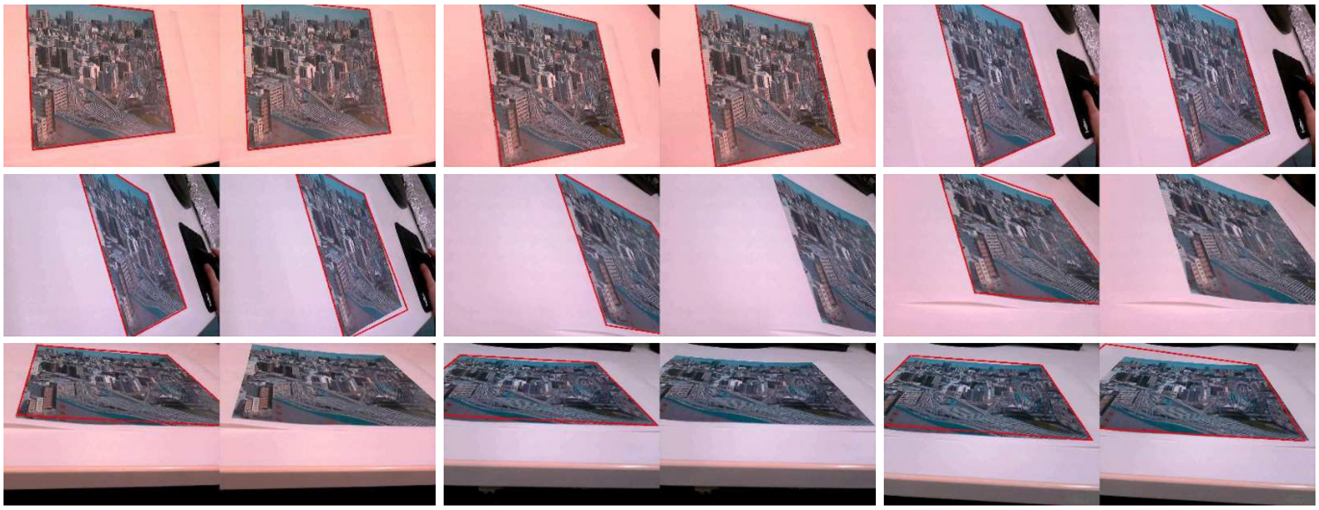


Fig. 6 Results of camera pose estimation using our method and VGL [8] using some evaluation frames. Left: our method, right: results from VGL [8].

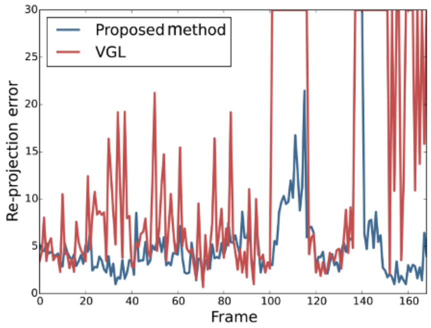


Fig. 7 Re-projection errors.

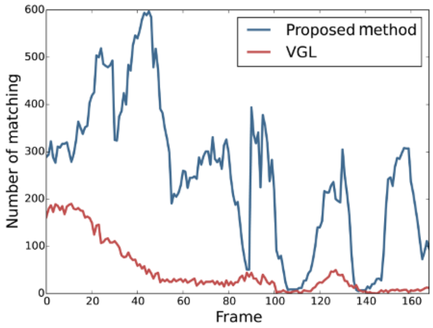


Fig. 8 Number of matches.

pattern but also the environment around it: the background, the lighting, and so on. However, the CNN can be only used in the same environment as the one in which the planar pattern was given and deep learning has been performed.

We next discuss processing time. Figure 9 shows the processing frame rate. Again, the overhead for viewpoint class estimation in the proposed method is sufficiently small.

3.2.3 Number of viewpoint classes

The number of viewpoint classes affects the results of camera pose estimation and the size of the databases. Therefore, we generated 100 test images with homographies and evaluated how the number of viewpoint classes affected the results for the method of learning based on generated viewpoints. Table 3 shows the re-projection error calculated by Eq. (5) and the database size when changing the number of viewpoint classes.

The re-projection error decreases with an increasing number of viewpoint classes since the input image and the matching image in the database become closer by splitting the viewpoint more finely. However, the size of the database is also increased as the number of generated databases is also increased. Thus, accuracy and size must be traded-off according to the particular application.

4 Conclusions

We have proposed a method for robust camera pose estimation using uncompressed descriptor databases generated for each viewpoint class. Our method classifies the viewpoint of each input image using a CNN that is trained by deep learning so that keypoints of the input image can be matched almost perfectly with the database. We gave two ways of generating these databases and preparing the images for deep learning. These methods have different applications. The first is robust in a changing environment, while the second allows the CNN to learn the entire environment around the planar pattern. We have experimentally confirmed that the number of keypoint matches was higher, and the accuracy of camera pose estimation was better, than with a conventional method.

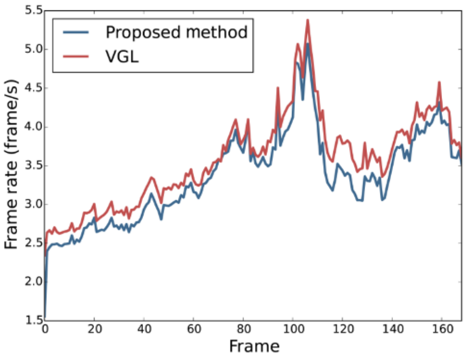


Fig. 9 Frame rate.

Table 3 Re-projection error and database size with respect to the number of viewpoint classes

Viewpoint classes	12	36	52
Re-projection error	1.21	0.90	0.81
Database size	16.6 MB	49.4 MB	67.9 MB

The application of our method to three-dimensional objects is our future work.

References

[1] Kato, H.; Billinghurst, M. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality, 85–94, 1999.

[2] Lee, T.; Hollerer, T. Hybrid feature tracking and user interaction for markerless augmented reality. In: Proceedings of IEEE Virtual Reality Conference, 145–152, 2008.

[3] Maidi, M.; Preda, M.; Le, V. H. Markerless tracking for mobile augmented reality. In: Proceedings of IEEE International Conference on Signal and Image Processing Applications, 301–306, 2011.

[4] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* Vol. 60, No. 2, 91–110, 2004.

[5] Mikolajczyk, K.; Schmid, C. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 27, No. 10, 1615–1630, 2005.

[6] Lepetit, V.; Fua, P. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 28, No. 9, 1465–1479, 2006.

[7] Breiman, L. Random forests. *Machine Learning* Vol. 45, No. 1, 5–32, 2001.

[8] Yoshida, T.; Saito, H.; Shimizu, M.; Taguchi, A. Stable keypoint recognition using viewpoint generative learning. In: Proceedings of the International Conference on Computer Vision Theory and Applications, Vol. 2, 310–315, 2013.

[9] Hartigan, J. A.; Wong, M. A. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* Vol. 28, No. 1, 100–108, 1979.

[10] Fukushima, K.; Miyake, S. Neocognitron: A new algorithm for pattern recognition tolerant of

deformations and shifts in position. *Pattern Recognition* Vol. 15, No. 6, 455–469, 1982.

[11] Hubel, D. H.; Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology* Vol. 160, No. 1, 106–154, 1962.

[12] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation* Vol. 1, No. 4, 541–551, 1989.

[13] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; Fei-Fei, L. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* Vol. 115, No. 3, 211–252, 2015.

[14] Agrawal, P.; Carreira, J.; Malik, J. Learning to see by moving. In: Proceedings of IEEE International Conference on Computer Vision, 37–45, 2015.

[15] Rumelhart, D. E.; Hintont, G. E.; Williams, R. J. Learning representations by back-propagating errors. *Nature* Vol. 323, 533–536, 1986.

[16] Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint* arXiv:1207.0580, 2012.

[17] Krizhevsky, A.; Sutskever, I.; Hinton, G. E. ImageNet classification with deep convolutional neural network. In: Proceedings of Advances in Neural Information Processing Systems, 1097–1105, 2012.

[18] Mikolajczyk, K.; Tuytelaars, T.; Schmid, C.; Zisserman, A.; Matas, J.; Schaffalitzky, F.; Kadir, T.; GooL, L. V. A comparison of affine region detectors. *International Journal of Computer Vision* Vol. 65, No. 1, 43–72, 2005.

[19] Fischler, M. A.; Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* Vol. 24, No. 6, 381–395, 1981.

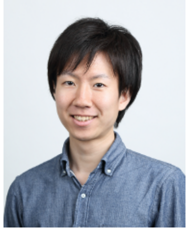
[20] Yu, G.; Morel, J.-M. ASIFT: An algorithm for fully affine invariant comparison. *Image Processing On Line* Vol. 1, 1–28, 2011.

[21] Ozuysal, M.; Calonder, M.; Lepetit, V.; Fua, P. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 32, No. 3, 448–461, 2009.

[22] Tokui, S.; Oono, K.; Hido, S.; Clayton, J. Chainer: A next-generation open source framework for deep learning. In: Proceedings of Workshop on Machine Learning Systems (LearningSys) in the 29th Annual Conference on Neural Information Processing Systems, 2015.

[23] Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv preprint* arXiv:1312.4400, 2013.

[24] Alcantarilla, P. F.; Nuevo, J.; Bartoli, A. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: Proceedings of British Machine Vision Conference, 13.1–13.11, 2013.



Yoshikatsu Nakajima received his B.E. degree in information and computer science from Keio University, Japan, in 2016. Since 2016, he has been a master student in the Department of Science and Technology at Keio University, Japan. His research interests include augmented reality, SLAM, object recognition, and computer vision.

Hideo Saito received his Ph.D. degree in electrical engineering from Keio University, Japan, in 1992. Since then, he has been on the Faculty of Science and Technology, Keio University. From 1997 to 1999, he joined the Virtualized Reality Project in the Robotics Institute, Carnegie Mellon University as a visiting researcher. Since 2006, he has been a full professor in the Department of Information and Computer Science, Keio University. His recent activities for academic conferences include being Program Chair of ACCV2014, a General Chair of ISMAR2015, and a Program Chair of ISMAR2016. His research interests include computer vision and pattern recognition, and their applications to augmented reality, virtual reality, and human robotics interaction.