Corner Detection Using Random Forests

Shubham Pachori, Kshitij Singh and Shanmuganathan Raman

Abstract We present a fast algorithm for corner detection, exploiting the local features (i.e. intensities of neighbourhood pixels) around a pixel. The proposed method is simple to implement but is efficient enough to give results comparable to that of the state-of-the-art corner detectors. The algorithm is shown to detect corners in a given image using a learning-based framework. The algorithm simply takes the differences of the intensities of candidate pixel and pixels around its neighbourhood and processes them further to make the similar pixels look even more similar and distinct pixels even more distinct. This task is achieved by effectively training a random forest in order to classify whether the candidate pixel is a corner or not. We compare the results with several state-of-the-art techniques for corner detection and show the effectiveness of the proposed method.

Keywords Feature extraction · Corner detection · Random forests

1 Introduction

Many computer vision and image processing tasks require highly accurate localization of corners along with fast processing. With these needs, many recent works have been proposed to address the corner detection problem. However, despite the increasing computational speed, many state-of-the-art corner detectors still leave a little time for processing live video frames at the capture rate. This motivates the need for faster algorithms for corner detection. Our method tackles the two key issues

S. Pachori (✉) · K. Singh · S. Raman
Electrical Engineering, Indian Institute of Technology Gandhinagar,
Gandhinagar, Ahmedabad, India
e-mail: shubham_pachori@iitgn.ac.in

K. Singh
e-mail: kshitij.singh@iitgn.ac.in

S. Raman
e-mail: shanmuga@iitgn.ac.in

B. Raman et al. (eds.), *Proceedings of International Conference on Computer Vision and Image Processing*, Advances in Intelligent Systems and Computing 459,
DOI 10.1007/978-981-10-2104-6_49

of corner detection—high accuracy and computational time. Apart from these, the algorithm is generic which can be used to detect corners in a given image of any natural scene.

We propose a method to detect corners trained on random forests on a set of corners obtained from training images. We use the trained random forests in order to detect corners in a new image. We show that the proposed approach is effective in the detection of corners in natural images. The algorithm could be used in a variety of computer vision tasks which rely on corner detection such as object recognition, image registration, segmentation, to name a few.

The primary contributions of this work are listed below.

1. A random forest-based learning framework for fast and accurate corner detection.
2. A simple and efficient way to obtain the feature descriptors for training images and the test image.

The rest of the paper is organized as below. In Sect. 2, we briefly review different major approaches for corner detection attempted in the past. In Sect. 3, we discuss the proposed method for corner detection. In Sect. 3.2, we present and compare our results with the state-of-the-art methods proposed earlier. In Sect. 4, we conclude our paper with directions for extending this work.

2 Related Work

The corner detection algorithms proposed in the past could be broadly divided into three main categories: (a) contour-based detectors, (b) model-based detectors and (c) intensity-based detectors as described in [1]. A more detailed survey is presented in [2]. Intensity-based corner detectors rely on the grey-level information of the neighbourhoods of pixels in order to detect the corners. Moravec gave the idea of defining points where a high intensity variation occurs in every direction as points of interest [3]. Harris and Stephens introduced a rotation invariant corner detector [4]. But, Harris corner detector gave poor results while detecting higher order corners as shown in [5]. Mikolajczyk and Schmid presented a novel technique to detect corners invariant to scale and affine transformation [6]. The calculation of first-order and second-order derivatives is computationally expensive. Moreover, the second-order derivatives are very sensitive to noise. SUSAN, proposed by Smith and Brady, uses a mask 'USAN' and compares the brightness corresponding to the nucleus of the mask, with the intensity of each pixel within the mask [7]. Corners in the image are represented by the local minima of the USAN map. Rosten and Drummond put forward the FAST algorithm which uses machine learning technique for corner detection [8]. It outperforms the previously proposed methods in computation time and high repeatability but leads to bad accuracy, giving some responses even along the edges. To improve this detection technique AGAST was proposed by Mair et al. [9]. This method apart from being less computationally expensive gives high performance for arbitrary environments. Later, Rosten et al. came with a new version of their early proposed FAST, called FASTER, which increased the accuracy but with the slightly increased computational time [10]. Jinhui et al. presented a double circle mask and presented their intuition behind the working of the double mask in the noisy environments [11]. We follow their procedure and use a double mask in our corner detection framework.

3 Proposed Approach

3.1 Motivation for Random Forest

Classification forest [12] is built from an ensemble of classification trees which form a class of non-parametric classifiers [13]. Each tree aims to achieve the classification task by splitting the node into multiple nodes based on some decision criterion. The splitting is performed at each internal (non-leaf) node based on problem-specific metrics for achieving the best splits. Generally, we do not make any assumption about the relationships between the predictor variables and dependent variables in the classification decision trees. Terminal nodes contain responses which classify the candidate into different classes, given its attributes. In this paper, we use information gain as the metric to make a split in each tree. Constructing a single decision tree has been found not giving desired results in most practical cases. This is because even small perturbations and noise in the data can cause changes in the results [14]. The key motivation for using random forest is to reduce the degree of correlation between different trees in forest, which then ultimately leads to a better generalization. We would like to train a random forest using corners detected in a set of natural images in order to detect corners in a given new natural image.

3.2 Random Forest Training and Corner Detection

We propose the usage of a double ring circle structure to be the feature descriptor as proposed in [15]. The intuition behind using this arrangement is that if there is a corner then there is a continuous change in its intensity values rather than abrupt changes. Hence, we could ignore the nearest 8-pixel neighbourhood circle around the candidate pixel. We could have also used the same three-circle mask as proposed in FASTER [10]. But we could remove the middle circle in those three consecutive circles, without much loss in performance due to the same reason. The two alternate circles are sufficient enough to classify a candidate pixel as corner or not. A double circle is also robust to noise as is shown in [15]. Therefore, we propose to use the circular mask as shown in Fig. 1 for extracting the feature descriptors. Let the intensity of the pixel at the nucleus of the mask be denoted by I_c and intensity of pixels within the mask be denoted by $I(x, y)$. The corner locations in the images are extracted using the Harris corner detector [4].

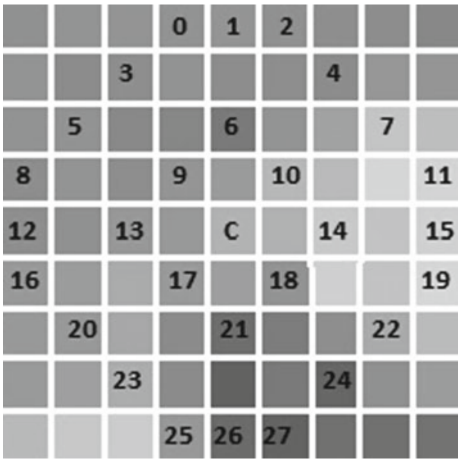

Fig. 1 The mask used by the proposed approach for feature description. The candidate pixel is marked by C

We then take the difference between the intensity value of the candidate pixel and the pixels of the double circular mask, $I_c - I(x, y)$ and form a vector $\bar{d}$, consisting of elements, $d_i, i = 0, 1, \ldots 27$. Rather than generically grouping them into darker, similar and lighter pixels as done in some of the previous works, we rather feed these values into a function as shown in Fig. 2a or b. The idea is to make the similar pixels look more similar and different pixels look more different. Any function which looks similar to these functions could do the job for us. One such function could be built using the hyperbolic tangent function. The equation of the function used by us could be written as:

$$\hat{d}_i = a(\tan\mathrm{h}(b \times d_i + c)) + a(\tan\mathrm{h}(b \times d_i - c)) \tag{1}$$

where a, b, and c are real constants.

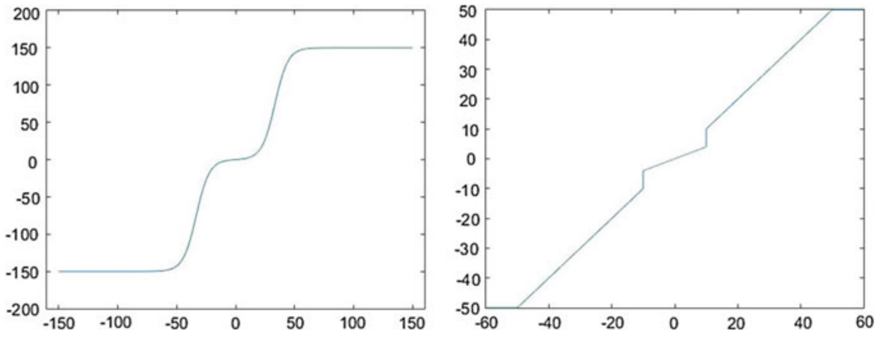

Fig. 2 Here d_i and $\hat{d}_i$ are in x- and y- axes, respectively. **a** Hyperbolic tangent function shown in Eq. 1, and **b** function built using ReLU [16] shown in Eq. 2

But the function built using tanh is computationally expensive. Therefore we resort to a modified version of the function ReLU (Rectified Linear Unit) as proposed in [16] which is defined as $\max(0, d_i)$. The function used by us built on ReLU is:

$$\hat{d}_i = \begin{cases} b \times d_i, d_i \in [-a, +a] \\ \min(-a + \max(0, d_i + a), a), \textit{otherwise} \end{cases} \tag{2}$$

where a and b are constants.

The first motivation behind this kind of learning is that small changes in intensities which are sufficient enough to classify a pixel as corner could be detected. Therefore, we have not defined some global threshold to group the pixels into the classes of darker, similar and lighter. Second, by not grouping the pixel values in these classes, we have not tried to make the trees learn by the class categories (similar, darker and lighter) as this method was observed to give poor results. The reason may be that dividing the neighbour pixels into three or five classes would have probably over-fitted the data as the training examples are huge and would have got the trees to memorize the pattern. The difference between these two methods will be discussed in the next section through results. The values obtain after processing the difference of intensities using the function in Eq. 2 would then serve as the final feature vector for the training set used to train the forest. Given a new image, we perform the same operation to extract the feature descriptors at all the pixel locations and mark the corners depending on the responses from the trained random forest. The decision is made for a given pixel to be a corner based on the average corner probability of the multiple tree outputs in the random forest. The complete procedure is described in Algorithm 1.

Algorithm 1 Corner Detection using Random Forest

```
1: Step 1: Detect corners in the N training images using Harris corner detection [4].
2: Step 2 : Train a random forest (with k trees) using the feature descriptors (shown in Fig. 1)
       around the detected corners and non-corner pixels.
3: Step 3: Extract feature descriptors from the test image using the pattern shown in Fig. 1.
4: Step 4: Use the trained random forest to detect corners in the test image.
```

We trained our random forests on an Intel i7-3770 processor at 3.40 GHz, with 8GB installed memory (RAM) on 64 bit Windows operating system in Python. We used OpenCV-Python library for image processing tasks and for comparisons between different methods of corner detection. We fixed the value of a and b in Eq. 2 to be 50 and 0.3, respectively, for the work. We compare the results obtained by our method using two cases. Case (i): we do not divide the pixels based on their relative intensities and Case (ii): we classify them as darker (0), similar (1) and lighter (2) as shown in Eq. 3.

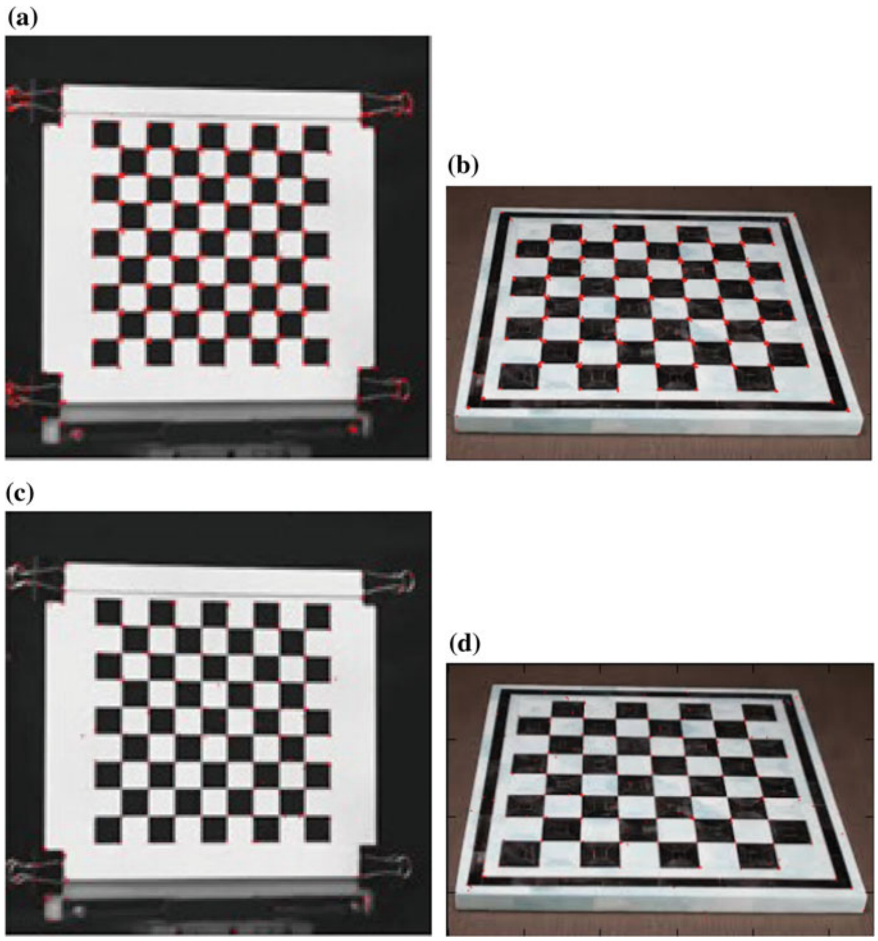

Fig. 3 **a** and **b** are the images obtained by case (i), **c** and **d** are the images obtained by case (ii)

$$d_i = \begin{cases} 0, & I_c - I(x, y) < -10 \\ 1, & -10 < I_c - I(x, y) < 10 \\ 2, & I_c - I(x, y) > 10 \end{cases} \tag{3}$$

For this comparison, we obtained training data from 35 images and the random forest comprised 15 trees. The results obtained are shown in Fig. 3. It can be observed that case (i) leads to better corner detection compared to that of case (ii). Therefore, we use case (i) to compare with other state-of-the-art methods.

For comparison, the number of trees trained was chosen to be 15 without any pruning of leaves. We created the training set from each and every pixel of 52 natural images. The training of the random forest took around 5 hours. We compare the results of our algorithm with that of Harris detector [4], SUSAN Corner Detector [7] and FAST-9 detector [8]. Average time taken by the methods for a typical 256×256 image were 0.07 s, 7.98 s and 6.86 s respectively. Harris corner detector is faster due to the inbuilt OpenCV-Python function. The best time taken by our method to compute the feature vector on the same image while using the proposed function built using Eq. 2 was 6.14 s and time taken to predict the corners in image, via 15 trees, was 0.3 s. It is much faster than the proposed function built on tanh function in Eq. 1 which took 31.3 s to compute the feature vector without any decrease in the accuracy. The time taken to compute the feature vector without using the proposed functions reduced to 1.12 s.

The images (none of which were trained) shown in Fig. 4 depict the accuracy of our algorithm. Harris corner detector still outperforms the other approaches in terms

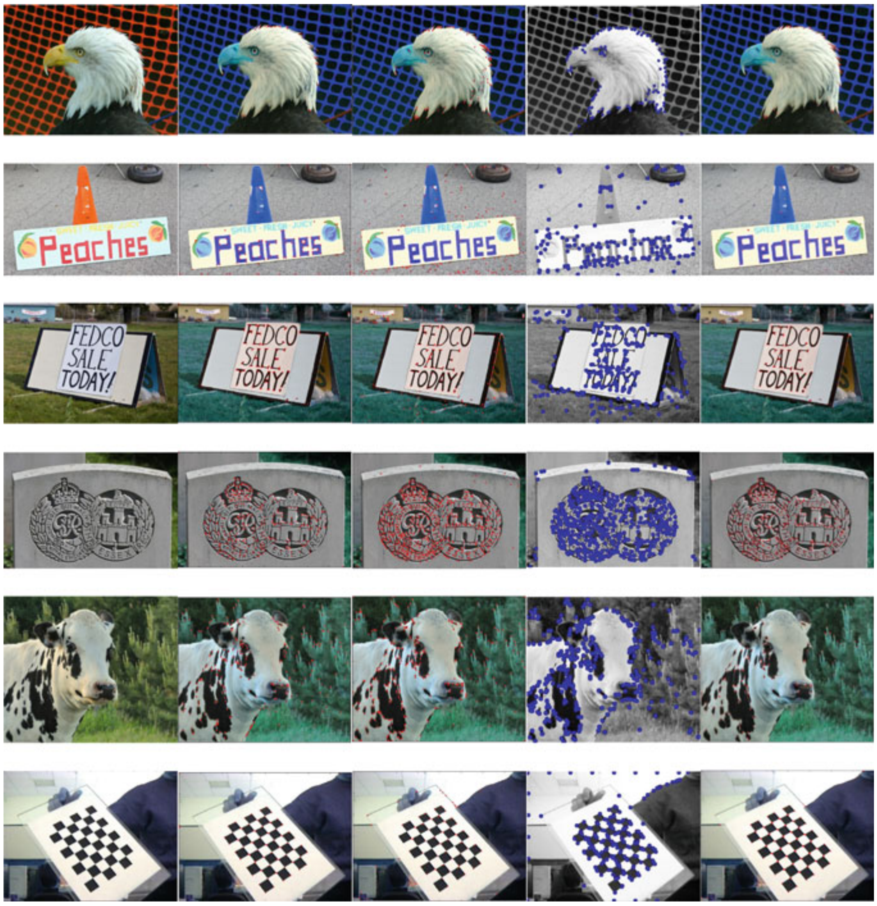

Fig. 4 First column—original images, corners detected using: second column—Harris corner detector [4], third column—FAST corner detector [8] fourth column—SUSAN [7], fifth column—proposed method

of accuracy. FAST corner detector showed poor results in terms of accuracy giving some points along the edges as could be seen in images in fifth and third rows of third column. SUSAN detected some extra potential non-corner points as corners. Our approach, though giving less number of corner points, gives nearly the same results as obtained from the Harris corner detector.

4 Conclusion

We have presented a fast learning-based method to detect the corners, whose performance is comparable to that of the state-of-the-art corner detection techniques. The choice of learning multiple decision trees using pre-detected corners along with the feature descriptor enables us to achieve this task. We are able to achieve high degree of accuracy in terms of detected corners using the proposed random forest framework. With very less number of trees and training images, we are able to detect corners in a given new image. In future, we would like to exploit random forests to detect the scale-invariant interest points in a given image and estimate the feature descriptors at these interest points. We would like to accelerate the framework proposed using GPU for various computer vision tasks. We believe that this framework will lead to increased interest in the research of learning-based corner detectors suitable for various computer vision applications.

References

1. A. Dutta, A. Kar, and B. Chatterji, Corner detection algorithms for digital images in last three decades, IETE Technical Review, vol. 25, no. 3, pp. 123133, 2008.
2. T. Tuytelaars and K. Mikolajczyk, Local invariant feature detectors: a survey, Foundations and Trends in Computer Graphics and Vision, vol. 3, no. 3, pp. 177280, 2008.
3. H. P. Moravec, Obstacle avoidance and navigation in the real world by a seeing robot rover. DTIC Document, Tech. Rep., 1980.
4. C. Harris and M. Stephens, A combined corner and edge detector. in Alvey vision conference, vol. 15. Citeseer, 1988, p. 50.
5. P.-L. Shui and W.-C. Zhang, Corner detection and classification using anisotropic directional derivative representations, Image Processing, IEEE Transactions on, vol. 22, no. 8, pp. 3204 3218, 2013.
6. K. Mikolajczyk and C. Schmid, Scale and affine invariant interest point detectors, International journal of computer vision, vol. 60, no. 1, pp. 6386, 2004.
7. S. M. Smith and J. M. Brady, Susan a new approach to low level image processing, International journal of computer vision, vol. 23, no. 1, pp. 4578, 1997.
8. E. Rosten and T. Drummond, Machine learning for high-speed corner detection, in Computer VisionECCV 2006. Springer, 2006, pp. 430 443.
9. E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, Adaptive and generic corner detection based on the accelerated segment test, in Computer Vision ECCV 2010. Springer, 2010, pp. 183196.

10. E. Rosten, R. Porter, and T. Drummond, Faster and better: A machine learning approach to corner detection, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 32, no. 1, pp. 105119, 2010.
11. J. Lan and M. Zhang, Fast and robust corner detector based on doublecircle mask, Optical Engineering, vol. 49, no. 12, pp. 127204127 204, 2010.
12. L. Breiman, Random forests, Machine learning, vol. 45, no. 1, pp. 532, 2001.
13. L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, Classification and regression trees. CRC press, 1984.
14. L. Breiman, Bagging predictors, Machine learning, vol. 24, no. 2, pp. 123140, 1996.
15. M. Trajkovic and M. Hedley, Fast corner detection, Image and vision computing, vol. 16, no. 2, pp. 7587, 1998.
16. V. Nair and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807814.