2013 12th International Conference on Document Analysis and Recognition

An Anytime Algorithm for Camera-Based Character Recognition

Takuya Kobayashi, Masakazu Iwamura, Takahiro Matsuda and Koichi Kise
Graduate School of Engineering, Osaka Prefecture University
{kobayashi, matsuda}@m.cs.osakafu-u.ac.jp, {masa, kise}@cs.osakafu-u.ac.jp

***Abstract*—In a scene image, some characters are difficult to recognize and some others are recognized easily. Such difficult characters usually make the processing time long while easy characters are recognized in a short time. In this paper, we propose a system which recognizes each character with a proper cost for the difficulty. Through the process, easy characters are recognized early and difficult ones are recognized late. This is a desired property of an anytime algorithm that the recognition accuracy does not decrease as the time increases. In order to realize it, we propose a method which splits the recognition process into several times and accumulates the recognition results and extracted features. We also discuss what is required to realize the anytime algorithm for the scene character recognition task. Experiments reveal that the proposed method obtains recognition results of easy characters earlier than the conventional method.**

Keywords*—scene character recognition; anytime algorithm; local feature; ASIFT; tracking; video input;

I. Introduction

As smart phones became more and more popular, many applications of scene character recognition appeared (e.g., Goggle Goggles, Evernote and Utsushite Honyaku from NTT Docomo). In order to improve their performance, realizing a fast and accurate character recognition system is an important task. We proposed a character recognition method using arrangements of local features [1]. The method is robust to perspective distortions thanks to an affine invariant local feature [2]. It is also robust to text layout changes such as characters printed on a curved line. The most effective point to use local features is the robustness to complex backgrounds. Existing segmentation-based methods fail to recognize characters when some parts of a character have a similar color with the background because those methods usually segment each character using the color contrast between the characters and the background. However, local features are extracted from many parts of a character and extracted features from the remaining parts can work well in such a case.

Though the method has such desirable properties, it is still far from real-time processing. What is worse, there is a limit to speed up the method because it is difficult to reduce the computational cost of the feature extraction process. Generally, the recognition accuracy and the processing speed depend on complexity of the background, and the similarity of characters and so on. As a result, some characters are difficult to recognize though the others can be recognized easily. We define easy characters as characters recognized in a short time and difficult ones are the opposite. The overall processing time is long because of the difficult characters.

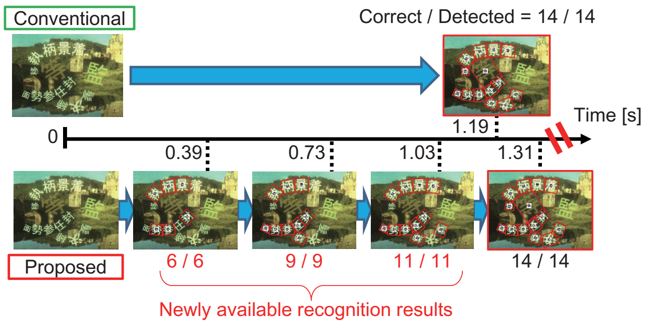


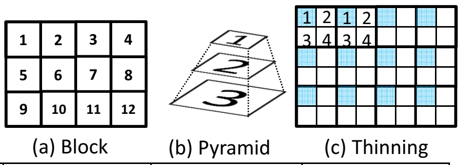
Fig. 1. An experimental result showing an overview of the proposed method. The common query image is shown in the leftmost position. The conventional method took 1.19 seconds to detect and recognize 14 characters in the query image with a complex background. On the other hand, the proposed method recognizes those characters in a step-by-step manner. 6 characters are detected and recognized in 0.39 seconds. Then, 9 characters are in 0.73 seconds, 11 characters are in 1.03 seconds, consequently, 14 characters are detected and recognized in 1.31 seconds. Though the proposed method took 0.12 seconds more than the conventional method to obtain the final result, some intermediate recognition results were obtained much earlier than the recognition result of the conventional method. Unlike the conventional methods, the proposed method satisfies some properties of an anytime algorithm. This enables us to obtain recognition results in the required amount of computational time depending on the difficulty of the characters.

In this paper, we propose a character recognition method which recognizes each character with a proper cost for the difficulty. We show an overview of it with an experimental result in Fig. 1. The idea is to split the recognition process into several times and accumulate the recognition results and extracted features. Through the process, easy characters are recognized early and the difficult ones are recognized late.

The proposed method has some desirable properties of an anytime algorithm [3]. An example is that the recognition accuracy does not decrease as the processing time increases. Therefore the user can stop the process when the user obtains sufficient results. We also discuss how we define the anytime algorithm for the scene character recognition task and what type of recognition method is preferable to realize it.

II. How to Realize an Anytime Algorithm for Character Recognition Methods

In this section, we discuss how we define the anytime algorithm for the scene character recognition task and how to realize it. First, we introduce the anytime algorithm. According to [3], a method needs to own some properties so that it is regarded as an anytime algorithm. In this paper, we take into account the two properties below.



	(a) Block	(b) Pyramid	(c) Thinning
Texture-based	×	○	○
Connected component-based	×	×	×
Local feature-based	×	○	○

Fig. 2. Relationship between three types of feature extraction methods and three methods to split the recognition process to realize an anytime algorithm. (a) means that a query image is divided into several blocks and the whole recognition process is sequentially applied to each of them. (b) is a coarse-to-fine approach that features are extracted from images with changing the resolution. (c) is to thin the pixels evenly in an image used for the feature extraction.

- Monotonicity: the quality of the results does not decrease as the computational time increases.
- Interruptibility: the process can be interrupted at any time and the user can get some results at the time.

Ideally the method can measure how well the recognition result reaches to the best result (Measurable quality). However, satisfying this requirement is difficult for character recognition tasks because the number of characters in query images is unknown until they are recognized. Thus we regard a character recognition method as the anytime algorithm if it has at least the two properties above. The basic idea to achieve them is to split the whole recognition process into several times and accumulate the recognition results and extracted features.

A problem here is whether the whole process of a character recognition including preprocessing, feature extraction and recognition step is splittable in a suitable way or not. It is obvious that the process-wise approach, where a process (e.g., recognition) starts after the previous process (e.g., feature extraction) completely finishes such like [4], does not resolve the problem. That is, splitting each process is required. It is noteworthy that calculating something (e.g., features) which is not immediately used in the succeeding process (e.g., recognition process) is inefficient for an anytime algorithm. Thus, how to split the first process, i.e., feature extraction, is the key issue. Figure 2 shows three representative feature extraction approaches and their capability to split the process in three ways. First of all, (a) is inappropriate because the recognition results are expected to be given from everywhere in the query image. By contrast, (b) and (c) are feasible because features are extracted from all over the image. Now we compare the three types of the character recognition methods. While texture-based methods (e.g., [5]) and local feature-based methods can be used to (b) and (c), connected components-based methods (e.g., [4], [6]) cannot be used to all the splitting methods. The reason is that the stability of connected components decreases when the characters get blurred in the pyramid in (b) or some parts of a character are missing in (c). In this paper, we select (c) as the splitting method as a trial because of the simple implementation. Besides, we use local features because the thinning process can be performed per pixel.

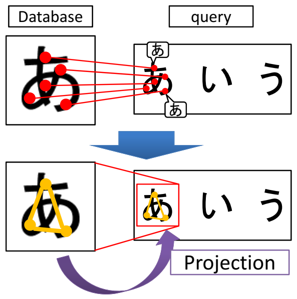


Fig. 3. An overview of the character recognition method using arrangements of local features. First, the extracted features from the query image are matched with those of reference images. Then the character region is projected to the query image by using the affine transformation matrix calculated from the arrangement of matched local features.

Another question is whether unreliable recognition results should be output. One may think that the results should be given regardless of the reliability at the time and their quality should be improved later. However, this reduces precision of the results which is important in the scene character recognition task. Thus we try to keep the precision rate as high as possible by using a reliability check for the recognized characters.

III. Character Recognition Method Using Local Features

In this section, we introduce our conventional character recognition method using arrangements of local features [1]. Figure 3 shows an overview of the method. The recognition process is mainly divided into three steps: feature extraction, feature matching, and character detection and recognition. We describe the detail of each step below.

In the feature extraction step, we use ASIFT [2] in order to detect feature points and describe feature vectors from a query image. ASIFT is an affine-invariant version of SIFT [7] and the features are robust to perspective distortions. Then, each feature vector in the query image is matched with feature vectors in the database. The database contains local features extracted from reference images. The pair with the shortest distance is regarded as corresponding. Here we use a fast approximate nearest neighbor search method called BDH [8]. After the process, each corresponding feature point in the query image is given the class label of the corresponding feature point in the database. The last step is character detection and recognition. The positions of the matched feature points are used to recognize characters. We use an assumption that each character region in the query image has a certain amount of corresponding feature points. Therefore, we search for such a region by looking into a close region around each feature point. However, this process cannot estimate the exact position of characters when same characters are printed adjacently in an image. Thus we use the arrangements of matched feature points in order to detect the exact region of each character. An affine transformation matrix is calculated from the positions of three pairs of feature points. By projecting the reference character region to the query image with the matrix, the exact region of the character is detected. In the process, the

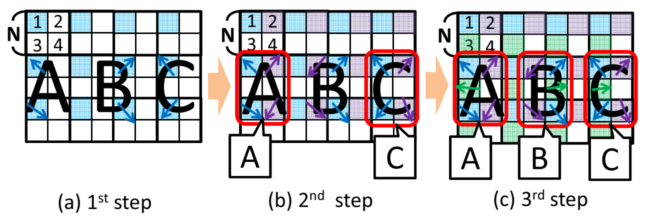


Fig. 4. An overview of the proposed method. The feature extraction step is split into N^2 times and features, represented by arrows, are extracted from different pixels in each step. The recognition results and the features are accumulated through the process. In this figure N = 2.

detection accuracy is improved by applying RANSAC [9] to the feature points in the character region. It randomly selects three pairs of corresponding features to compute an affine transformation matrix. The process is repeated R times and the best matrix is selected. Increase of R improves the recognition accuracy. Since RANSAC reduces the probability of misdetection, we can treat the character which passes through the process. Finally, each recognized character has a score based on the confidence ratio to the recognition result. The score is calculated by

$$\text{Score} = \frac{m_p}{\sqrt{r_p}}, \tag{1}$$

where m_p is the number of feature points inside the character region which has the corresponding character label. r_p is the number of feature points in the reference image of the character class. $\sqrt{r_p}$ is used to normalize the difference between the number of feature points extracted from each reference character. In case several characters are recognized in almost same region, we group the characters and the one with the highest score is chosen as the recognition result.

IV. Proposed Method

In this section, we introduce the proposed method based on the character recognition method presented in Sec. 3. Figure 4 shows an outline of the proposed method. The main idea is to split the recognition process into several times and accumulate the recognition results and extracted features. We describe the detail of the proposed method below.

Given a query image, the proposed method splits the image into a number of cells. The feature extraction step is split into N^2 times and features are extracted from different pixels in each step. Thus the recognition accuracy and the processing time for each step are changed by the number of N. By accumulating the recognition results through the process, the recognized characters basically increase as the processing time increases. Theoretically the recognition accuracy reaches to the same recognition accuracy. In order to reduce the computational cost, we introduce several ideas to the process. The first is to skip the recognition process on the character regions which are already recognized in the previous steps. Through the process, accumulated local features increase and the computational cost also increases. We can reduce the computational cost with the skipping. In order to skip only the characters recognized with a high confidence, we set a threshold T for the score given by Eq. (1). If the calculated score is lower than T, the recognition process is not skipped in the region.

V. Experiment

In this section, we show the experimental results. In order to evaluate the effectiveness of the proposed method, we compared the recognition accuracy and the processing time with the conventional method [1]. The conventional and proposed methods shared two parameters of BDH and R, which were determined based on preliminary experiments. R was set to be 25 for both methods. In addition, T was set to be 0.3 for the proposed method. In the experiments, we employed 71 categories of Hiragana, 71 categories of Katakana and 1,945 categories of Kanji (Chinese character) in MS Gothic font for reference characters with the same condition as [1]. The resolution of the camera used in the experiments was 640 $\times$ 480. All experiments were performed on a computer with Intel Core i7 3.50GHz CPU and 8GB memory. We prepared 4 query images for each character category: 2 different set of characters were put on 2 different backgrounds. Figure 5 shows some examples of the queries. The size of characters varied from 50 to 150 pixels for each side of a square bounding box.

Figure 6 shows some recognition results of the proposed method. As already discussed in [1], the recognition accuracy basically depends on the complexity of character shapes. Many local features are extracted from complex shapes of Kanji characters, which contributed to achieve high recognition accuracy. Figure 7 shows the recall, precision and F-measure for each character category. The results for each character category were averaged over the different 4 types of the query images. As shown in Fig. 7, the proposed method output recognition results earlier than the conventional method, while the total computational time of the proposed method increased. The parameter of $N = 2$ performed best in the proposed method. There are two reasons for this. With regard to computational cost, obtaining recognition results takes a constant time. Thus as N increases, it takes more time because the number of recognition processes increases. With regard to recognition performance, outputting the recognition result at an early stage with a fewer number of local features is more risky than doing it at a latter stage with a more number of local features. A better rejection strategy may improve this.

Next, we consider the reason why the precision rate decreased in some parameters of the proposed method. There are two main reasons. First, because some pairs of characters in Hiragana and Katakana are very similar, such characters were sometimes recognized as the wrong one. In the center of Fig. 6 (b), a character “” was recognized wrongly as “”. Since such pairs are quite difficult to distinguish, one solution is to recognize them in the following process such as using contexts. Another reason is related to the problem of such similar characters to some extent. Some characters in a query image were sometimes misrecognized in early steps. This problem occurred many times when $N = 3, 4$ because the more we split the feature extraction process, the less feature points we can extract in early steps. The same problem occurred when some parts of a character have a similar color with its background. In the case, recognition results were sometimes given only from the region that the contrast was high (e.g., a big brown

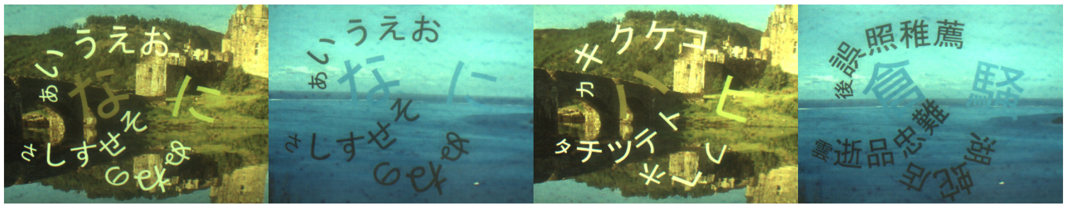
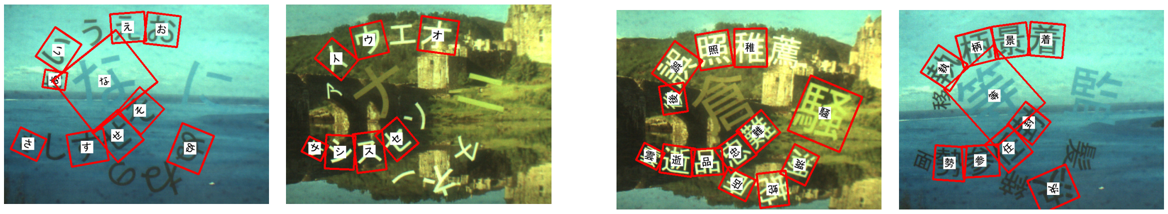

(a) Hiragana1 (b) Hiragana3 (c) Katakana2 (d) Kanji3

Fig. 5. Some examples of the query images. 15 characters were put on each image. We prepared 2 different set of characters and 2 different backgrounds for each character category.


(a) Hiragana3 (N=3)
Recall = 0.53 Precision = 1.00

(b) Katakana1 (N=2)
Recall = 0.60 Precision = 1.00

(c) Kanji1 (N=3)
Recall = 0.87 Precision = 1.00

(d) Kanji4 (N=4)
Recall = 0.87 Precision = 1.00

Fig. 6. Some recognition results of the proposed method. The results were given after N^2 times of steps. Red rectangles represent a character region of each recognized character and the recognition results are put on the center of each rectangle.

character in the middle of Figure 6 (c)). In order to solve this problem, we need a better strategy to judge if a recognition result is proper for the region or not.

VI. How to Support Video Input

Query images for the proposed methods are assumed to be still character images. However, we have already realized a system for video input. In this section we discuss what are required for it while its details and evaluation will be presented in another opportunity. An advantage of it is that many varieties of local features can be extracted from each video frame. Since stability of local features depends on noises and illumination changes, extracting stable local features from only one captured image is sometimes difficult. In particular, when the characters in an image have simple shapes, the number of stable features is quite few and recognizing such characters is very severe. Thus we accumulate the local features extracted from each frame in order to improve the recognition accuracy for such characters. In video input, the position of characters in each frame may gradually change and the position of extracted feature points also change. Therefore we cannot apply some functions of the proposed method presented in Sec. 4 directly to the video input. In order to solve it, we adjust several things of the method. First, we need to create a Gaussian pyramid in each frame for the feature extraction because the captured images vary in every frame. Besides, when we split the feature extraction process into N^2 times, we have to take into account the order of pixels from which local features are extracted because we preferably extract local features evenly from each cell. Second, we require a way to trace the movement of captured image sequence. We achieved it by using the KLT tracker which is a fast and robust tracking method [10]. It calculates a homography from the previous frame to the current frame. We use the homography to project and accumulate the recognized characters and the local features at the correct positions. At last, unlike a still image, the number of accumulated feature points increases as we capture many video frames. Thus we set a lifetime to each local feature to restrict the increase. The length of the lifetime varies depending on the number of feature points each reference character image has. Since characters with a simple shape has less features compared with complex ones, the length need to be longer. We confirmed the lifetime prevented the computational time from increasing.

VII. Conclusion

Generally some characters in a query image are recognized in a short time and some others take long. The difficult characters usually make the processing time long while the easy characters are recognized in a short time. The overall processing time is determined by the difficult characters. In this paper, we proposed a character recognition method which recognizes each character with a proper cost. The idea to realize it was to split the recognition process into several times and accumulate the recognition results and features. Through the process, easy characters are recognized early and the difficult ones are recognized late. This is one of the desirable properties of an anytime algorithm. We discussed how we define the anytime algorithm for character recognition tasks and how to realize it. As a result, we introduced a thinning-based approach to a local feature-based feature extraction method. We suggest that the proposed idea is applicable to not only a local feature-based feature extraction method but also a texture-based feature extraction method. In the experiment, we confirmed the proposed method obtained recognition results of easy characters earlier than the conventional method.

Our future work is to evaluate the performance of the pro-

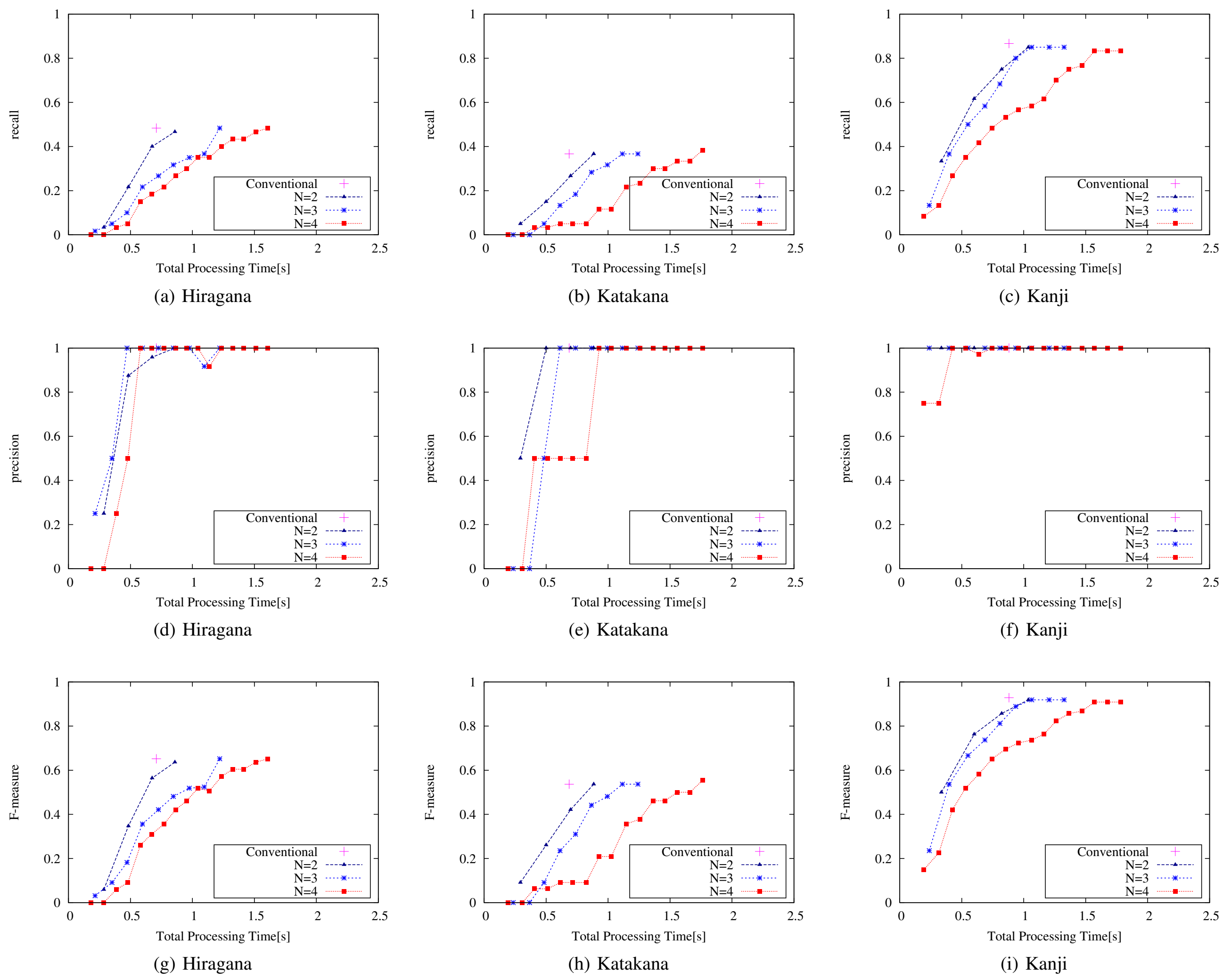


Fig. 7. Relationships between the computational time and recall (top), precision (middle) and F-measure (bottom) for each character category.

posed method with video input. It helps us recognize difficult characters by accumulating stable local features extracted from each video frame.

Acknowledgment

This work was supported in part by JST CREST project and JSPS KAKENHI Grant Number 25240028.

References

[1] M. Iwamura, T. Kobayashi, and K. Kise, “Recognition of multiple characters in a scene image using arrangement of local features,” *Proc. ICDAR*, pp. 1409–1413, 2011.

[2] J. Morel and G.Yu, “ASIFT: A new framework for fully affine invariant image comparison.” *SIAM Jour. on Imaging Sciences*, vol. 2, 2009.

[3] S. Zilberstein, “Using anytime algorithms in intelligent systems,” *AI magazine*, vol. 17, no. 3, pp. 73–83, 1996.

[4] L. Neumann and J. Matas, “Real-time scene text localization and recognition,” in *Proc. CVPR*, 2012.

[5] J.-J. Lee, P.-H. Lee, S.-W. Lee, A. L. Yuille, and C. Koch, “Adaboost for text detection in natural scene,” in *Proc. ICDAR*, 2011, pp. 429–434.

[6] C. Yao, Z. Tu, and Y. Ma, “Detecting texts of arbitrary orientations in natural images,” in *Proc. CVPR*, 2012.

[7] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[8] T. Sato, M. Iwamura, and K. Kise, “Fast and memory efficient approximate nearest neighbor search with distance estimation based on space indexing,” IEICE Technical Report, PRMU2012-142, 2013, in Japanese.

[9] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] J. Shi and C. Tomasi, “Good features to track,” *Proc. CVPR*, pp. 593–600, 1994.