EVALUATION OF SPEED AND ACCURACY FOR COMPARISON OF TEXTURE CLASSIFICATION IMPLEMENTATION ON EMBEDDED PLATFORM

[1]Jing Yi Tou, [1]Kenny Kuan Yew Khoo, [1]Yong Haur Tay, [2]Phooi Yee Lau

[1]Computer Vision and Intelligent Systems (CVIS) Group
Faculty of Information and Communication Technology
Universiti Tunku Abdul Rahman (UTAR)
Petaling Jaya, Malaysia
E-mail: tayyh@mail.utar.edu.my
[2]Instituto de Telecomunicações
Lisboa, Portugal
E-mail: laupy@lx.it.pt

ABSTRACT

Embedded systems are becoming more popular as many embedded platforms have become more affordable. It offers a compact solution for many different problems including computer vision applications. Texture classification can be used to solve various problems, and implementing it in embedded platforms will help in deploying these applications into the market. This paper proposes to deploy the texture classification algorithms onto the embedded computer vision (ECV) platform. Two algorithms are compared; grey level co-occurrence matrices (GLCM) and Gabor filters. Experimental results show that raw GLCM on MATLAB could achieves 50ms, being the fastest algorithm on the PC platform. Classification speed achieved on PC and ECV platform, in C, is 43ms and 3708ms respectively. Raw GLCM could achieve only 90.86% accuracy compared to the combination feature (GLCM and Gabor filters) at 91.06% accuracy. Overall, evaluating all results in terms of classification speed and accuracy, raw GLCM is more suitable to be implemented onto the ECV platform.

Keywords: Texture Classification, Computer Vision, Embedded System, Embedded Computer Vision

1. INTRODUCTION

Embedded computer vision (ECV) platform is a platform designed to deploy computer vision applications onto the embedded system [1]. The reduction of processor's size and cost, and increase in processing capability has enabled more applications to be deployed onto embedded systems. Embedded systems offer a few benefits, including compactness, portability, as well as the ability to be applied in various environments.

Texture classification is used for applications that involves texture like subjects, i.e. rock texture classification [2], wood species recognition [3,4] and face detection [5]. Wood species recognition for example is useful when used at actual site such as in the remote forest. On the other hand, embedding the system onto the ECV platform also ensures that the device can be installed in a fix location with less difficulty due to its compactness. However, there are some challenges that have to be solved during the deployment process. Embedded platforms generally have less processing power compared to the PC platform. For this purpose, the algorithms that are going to be implemented cannot be too complex; else, it will cause much delay in the classification process. Besides, embedded platforms, depending on implementation cost, may have a smaller memory capacity, which limits the number of training data that can be stored. Bear in mind that actual deployment may face much difficulty as both have different architecture. Our work, so far, has been concentrating in optimizing classification methods for wood recognition [4,6,7].

In this paper, our main objectives are:

- Implementation of GLCM and Gabor filters for the texture classification problem.
- Facilitate the deployment of the best algorithm onto the ECV platform.
- Performance comparison of the texture classification algorithms on the ECV platform – speed and accuracy.

Section 2 describes the algorithms tested in this paper. Section 3 describes the ECV platform and the application deployment process. Section 4 includes the experimental dataset and tools used. Section 5 analyses the experimental results. Section 6 concludes the paper.

2. METHODOLOGY

In this paper, the algorithms tested include the grey level co-occurrence matrices (GLCM) and Gabor filters along the combinations methods used in [6].

2.1 GLCM and Gabor Filters

2.1.1 GLCM

GLCM is a widely known texture classification technique since 1973 [8]. It remains to be useful to solve many different texture classification methods, including rock texture classification and wood species recognition.

The GLCM can be used to produce second-order textural features, such as contrast, homogeneity, energy, entropy

and correlation [6]. The parameters of the GLCM used in this paper includes the spatial distance $d \in \{1, 2, 3, 4, 5\}$ pixels, four orientations $o \in \{0°, 45°, 90°, 135°\}$ and number of grey level $G \in \{8, 16, 32, 64, 128, 256\}$ are used in the generation of GLCM. On the other hand, the GLCM itself can be used as feature other than the second-order textural features. For the raw GLCM, the GLCM is downsized by omitting values rather than averaging the values.

2.1.2 Gabor Filters

The Gabor filters are also known as Gabor wavelets. It is inspired by the concept of mammalian simple cortical cells [5]. The Gabor filters is represented by Equation (2.1) where x and y represent the pixel position in the spatial domain, ω^{ρ} represents the radial center frequency, θ represents the orientation and σ represents the standard deviation of the Gaussian function along the x- and y- axes where $\sigma_x = \sigma_y = \sigma$ [5] and the Gabor filter is illustrated in Fig. 1.

$$\Psi(x, y, \omega_0, \theta) = \frac{1}{2\pi\sigma^2} e^{-\left((x\cos\theta + y\sin\theta)^2 + (-x\sin\theta + y\cos\theta)^2\right)/2\sigma^2} \times \left[e^{i(\omega_0 x\cos\theta + \omega_0 y\sin\theta)} - e^{-\omega_0^2\sigma^2/2}\right] \quad (2.1)$$

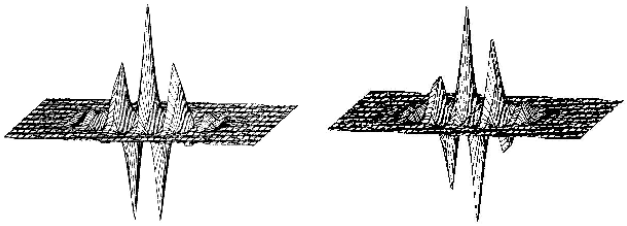

Fig. 1: Real part (left) and imaginary part (right) of a Gabor filter [9].

The Gabor filters are generated using different radial center frequencies and orientations. In this paper, three radial center frequencies and eight orientations are adopted to generate the Gabor filters [5].

A fast convolution is performed by applying fast Fourier transform (FFT), point-to-point multiplication and inverse fast Fourier transform (IFFT) [5]. Since the Gabor features is lying on a high-dimensional space, therefore it will cause higher difficulty in the classification stage. A down-sampling is performed on the Gabor features.

A principal component analysis (PCA) is further performed to reduce the dimensionality of the Gabor features. The singular value decomposition (SVD) is used to decompose the feature matrix into three smaller matrices which only one of them is chosen to produce the final features [6].

2.1.3 Combined Features

The GLCM features and Gabor features can be combined to be used as a set of features as a whole. The Gabor features are appended after the GLCM features [6].

2.2 Feature Normalization

The feature normalization is helpful to normalize the features to reduce the bias of certain features over the others. The normalization process will generate the distribution of the feature among all samples and chooses the value at the 1st and 99th percentile. This avoids the absolute maximum and minimum to be chosen as they might be outliers of the features. The normalized features $N(x)$ is shown in Equation (2.2) where x represents the original feature values, F_{min} represents the minimum feature value and F_{max} represents the maximum feature value [3].

$$N(x) = \frac{x - F_{min}}{F_{max} - F_{min}} \quad (2.2)$$

2.2 k-Nearest Neighbor (k-NN)

The k-NN is used as the classifier for the implementations in this paper. The nearest neighbor classifier compares the prompt sample with every training sample and chooses the k best training samples with the smallest Euclidean distances. The class with the highest number of nearest neighbor selected will be regarded as the winning class.

The disadvantage of the k-NN is the algorithm requires all the training samples to be stored in the memory and the comparison with all the training samples causes the speed to be slower as the number of training samples increases. On the other hand, if the training set is too small, it fails to represent the population and therefore the accuracy will decrease. In this paper, the k-NN used for the experiments is using the value of $k \in \{1, 2, 3, \ldots, 10\}$.

3. EMBEDDED SYSTEM

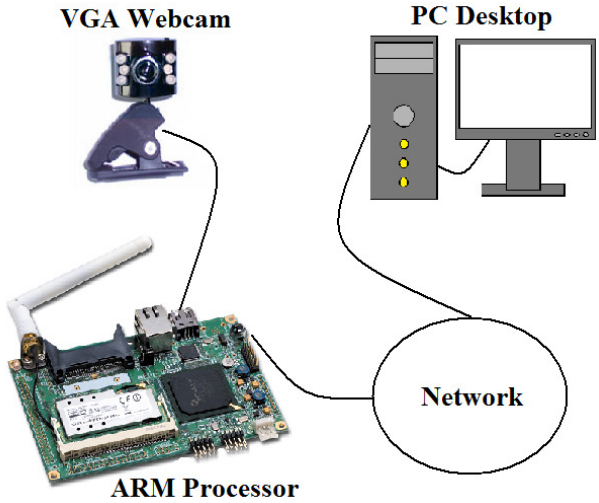

Fig. 2: Setup of ECV platform.

The ECV platform is an embedded platform using the ARM processing board which is running on the Linux operation system. The platform uses the Debian Linux distribution on the Linux 2.6 kernel. A VGA webcam with resolution of 320 × 240 pixels is used in the prototype of

the ECV platform. However, in this paper, current experiment uses offline images from the dataset to compute classification speed and accuracy, and the VGA webcam is not used. Since there are no output display devices and input buttons installed onto the ECV platform yet, the I/O communication is accomplished within a network. The architecture of the ECV platform is as shown in Fig. 2.

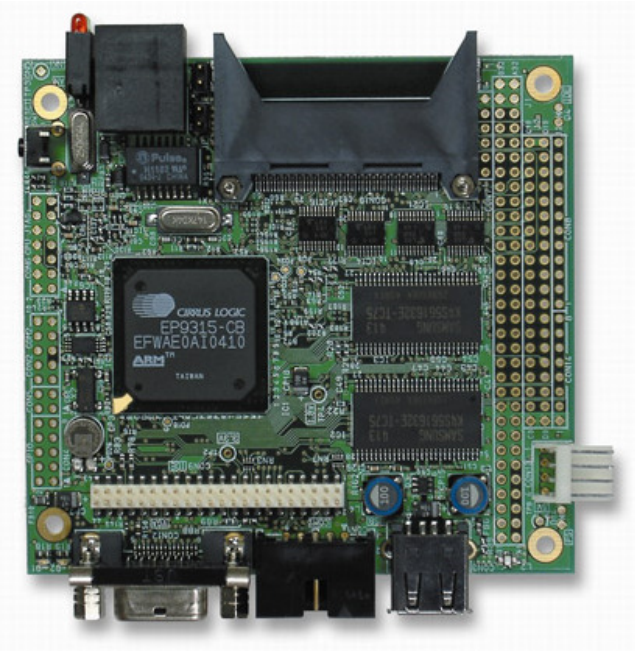

Fig. 3: ARM920T Processing Board.

Fig. 4: ARM926EJ-S Processing Board.

The two ARM processing boards that are used in this paper are the ARM920T and the ARM926EJ-S, shown in Fig. 3 and Fig. 4, respectively. The ARM920T has a Cirrus Logic EP9315 200MHz processor while the ARM926EJ-S has a Digi International NS9750B 200MHz processor. Both ARM processing boards has 64MB of RAM and 2GB of harddisk space.

The codes written in ANSI C are compiled on the ECV platform and the results are verified against the PC platform to ensure that they are performing the same work. Due to less computational power in the ECV platform, algorithms deployed must be computationally inexpensive, in order for it to perform in an acceptable speed.

4. EXPERIMENTAL MATERIALS

4.1 Dataset

The dataset used in this paper is including 32 textures from the Brodatz texture dataset [10] used in [6,7,11,12] as shown in Fig. 5.

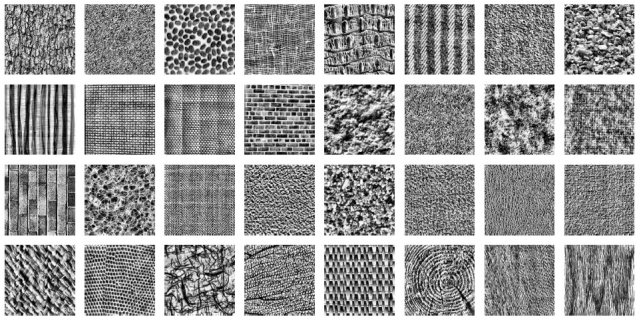

Fig. 5: 32 Brodatz textures from the Brodatz dataset [11].

Each texture has 16 separate partitions with samples of 64 × 64 pixels. Each of these partitions have four variants, which are the original image, scaled image, rotated image and the image both scaled and rotated. Eight of these partitions and their respective variants are randomly picked from each texture as the training samples while the others are used as the testing samples. For each experiment, ten random sets are picked according to the criteria above.

4.2 Development Tools

The experiments are performed using MATLAB. MATLAB is used as the development tool as it is a tool that allows rapid development to be accomplished. The Image Processing Toolbox and Bioinformatics Toolbox are used for the GLCM and k-NN algorithms.

The Microsoft Visual Studio.NET is used for the development of C programming codes as the ECV platform requires ANSI C programming codes.

5. RESULTS AND ANALYSIS

5.1 Experiment using Raw GLCM

In this experiment, the raw GLCM is used instead of using the second-order statistics. To reduce the size of the features, the GLCM is downsized to a 4 × 4 matrix. The results are shown in Table 1 where the horizontal bar represents the spatial distance and the vertical bar represents the number of grey level. The results are shown for the best value of *k* of the k-NN.

Table 1: Results of raw GLCM.

(%)	1	2	3	4	5
8	**90.86**	87.38	80.94	68.75	58.75
16	75.08	64.06	57.30	48.79	41.72
32	52.42	41.21	34.80	27.23	25.47
64	29.57	23.16	16.33	14.57	14.77
128	19.88	12.77	8.83	7.89	7.81
256	9.61	8.01	4.77	5.98	5.66

For the normalized raw GLCM, the results are shown in Table 2 where the horizontal bar represents the spatial distance and the vertical bar represents the number of grey level. The results are shown for the best value of k of the k-NN.

Table 2: Results of normalized raw GLCM.

(%)	1	2	3	4	5
8	**90.00**	87.62	80.86	66.76	56.13
16	71.64	66.13	55.59	47.19	39.22
32	35.55	33.87	29.65	24.02	17.46
64	15.08	15.04	12.81	10.47	9.38
128	7.77	8.01	5.94	5.35	5.31
256	4.57	4.92	4.34	4.65	3.91

The highest accuracy achieved is 90.86% by using 8 grey levels and spatial distance of 1 pixel. The higher the number of grey level is, the lower the accuracy. The accuracy is lower when the number of grey level is high because the larger GLCM is less homogenous compared to GLCMs at lower number of grey level and more information within the GLCM will be omitted from a larger GLCM.

The results for the normalized GLCM are worse than the original GLCM unlike other features such as GLCM features and Gabor features. This is so because some features are very small within the GLCM, normalization will maximize these unimportant features and therefore bringing down the accuracy.

5.1 Comparison of Speed and Accuracy

Comparison of classification speed for different algorithms on the PC platform helps to identify the potential algorithm that will be selected for the deployment onto the ECV platform. The methods involved in the performance comparison include methods that are previously implemented [6].

The algorithms are tested on an Intel T7500 Core 2 Duo 2.20GHz with 4GB of RAM – PC platform, using MATLAB. The computation time is calculated as a single process, from image acquisition to the generation of the result. The input image is an offline image selected from the testing set with the size of 64 × 64 pixels. The time duration is represented in millisecond (ms) which is the average time duration from ten runs of the classification process. The comparison of the classification time of different methods on both Windows XP and Linux Ubuntu and their classification accuracy are shown in Table 3.

Table 3: Comparison of classification time and accuracy for different operating systems using MATLAB.

	Windows XP (ms)	**Linux Ubuntu (ms)**	**Accuracy (%)**
GLCM features	51	54	85.73
Raw GLCM	50	50	90.86
Gabor features	901	1096	79.87
GLCM features + Gabor features	924	1113	91.06

The results for the combined feature of GLCM and Gabor filters is the highest at 91.06% but the time duration of a single process is more than 1 second. The fastest algorithm is the raw GLCM which only need 50ms while the accuracy is 90.86% which is only 0.2% lower than the previous method. Because the Gabor filters are much complex than the GLCM, with higher computation time and more effort, they are much slower. Overall, we suggest deploying raw GLCM on the ECV platform despite having slightly lower accuracy.

Since the ECV platform uses ANSI C codes, codes written in MATLAB are being rewritten in C language using Microsoft Visual Studio.NET. It is then re-tested on both the PC platform and ECV platform. The C codes presented here are compatible on both Windows and Linux platform. The comparison of the average computation time for the raw GLCM method is tested on different platforms and the results are shown in Table 4.

Table 4: Comparison of classification time on different platforms.

Platform	**Specification**	**Operating System**	**Time (ms)**
i686 PC Platform	Intel T7500 2.2GHz 4GB RAM	Linux Ubuntu 8.04	43
i686 PC Platform	Intel T7500 2.2GHz 2GB RAM	Debian Linux (virtual machine)	61
ARM920T ECV Platform	Cirrus Logic EP9315 200MHz 64MB RAM	Debian Linux	3708
ARM926EJ-S ECV Platform	Digi International NS9750B 200MHz 64MB RAM	Debian Linux	7342

Due to lower processing power, the processing time on the ECV platform is significantly less than running it on the PC platforms. Experimental results show that the time needed on the ECV platform is 86 times slower than the PC platform, using a 2.2GHz dual core processor. From the previous results shown in Table 3, we predict that implementing Gabor filters will consume too much time on the ECV platform, therefore, it is not suitable to be deployed in actual implementation. Keep in mind that the C code is not as optimized as the MATLAB codes, as

optimizing these codes were not our prior objectives.

6. CONCLUSION

Our work evaluates various texture classification algorithms implemented onto the ECV platform. Experimental results show that the raw GLCM could perform faster and have better performance on the PC platform although the accuracy is slightly lower than the combined feature of GLCM and Gabor filters. This is due to the fact that the combined feature of GLCM and Gabor filters performed much slower than the raw GLCM.

Although the combined feature of GLCM and Gabor filters were performing better than the raw GLCM itself, the computation of Gabor filters is much complex, and will take more time to process. Since the embedded systems has less processing power than normal PC, the combined feature is not suitable to be implemented onto the ECV platform, when compared to the raw GLCM.

The C programming codes can be further optimized to provide a faster implementation on the ECV platform as the self written C programming codes is less optimal than the MATLAB GLCM and k-NN functions. This provides space for improvement in the speed of the implementation.

As for future development, a classifier which store less information and performing less computation than the k-NN could be identify to speed-up the processing of the algorithm on the ECV platform. With a faster speed, real-time applications can be then deployed onto the ECV platform for actual site implementation.

7. ACKNOWLEDGEMENT

This research is partly funded by Malaysian MOSTI ScienceFund 01-02-11-SF0019.

8. REFERENCES

[1] K. K. Y. Khoo and Y. H. Tay, "A Preliminarily Study on Embedded Platforms for Computer Vision Applications," in *Regional Postgraduate Conference on Engineering and Science (RPCES)* pp. 183-189, 2006.

[2] M. Partio, B. Cramariuc, M. Gabbouj and A. Visa, "Rock texture retrieval using gray level co-occurrence matrix," in *Proceedings of 5th Nordic Signal Processing Symposium*, 2002.

[3] Y. L. Lew, "Design of an intelligent wood recognition system for the classification of tropical wood species," M. E. thesis, Universiti Teknologi Malaysia, Malaysia, 2005.

[4] J. Y. Tou, P. Y. Lau and Y. H. Tay, "Computer Vision-based Wood Recognition System," in *International Workshop of Advanced Image Technology (IWAIT)* pp. 197-202, 2007.

[5] W. H. Yap, M. Khalid and R. Yusof, "Face Verification with Gabor Representation and Support Vector Machines," in *IEEE Proc. of the First Asia International Conference on Modeling and Simulation*, 2007.

[6] J. Y. Tou, Y. H. Tay, P. Y. Lau, "Gabor Filters and Grey-level Co-occurrence Matrices in Texture Classification," in *MMU International Symposium on Information and Communications Technologies (M^2USIC)*, 2007.

[7] J. Y. Tou, Y. H. Tay, P. Y. Lau, "One-dimensional Grey-level Co-occurrence Matrices for Texture Classification," in *International Symposium on Information Technology (ITSIM)* Vol. 3. pp.1592-1597, 2008.

[8] R. M. Haralick, K. Shanmugam and I. Dinstein, "Textural Features for Image Classification," in *IEEE Transactions on Systems, Man. and Cybernatics* pp. 610-621, 1973.

[9] M. Nixon and A. Aguando, *Feature Extraction & Image Processing.* Butterworth-Heinemann Great Britain, 2002.

[10] P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* Dover New York, 1996.

[11] T. Ojala, M. Pietikainen, J. Kyllonen, "Gray level Cooccurrence Histograms via Learning Vector Quantization," in *Proc. 11th Scandinavian Conference on Image Analysis* pp. 103-108, 1999.

[12] T. Ojala, K. Valkealahti and M. Pietikainen, "Texture Discrimination with Multidimensional Distributions of Signed Gray Level Differences," in *Pattern Recognition 34*, pp. 727-739, 2001.