Time of Analog Clocks and Machine Learning

COSC 6342 Final Project
Alexander Hebert

Introduction

Interpreting images of common objects is not a difficult task for most people. While for machines, recognizing objects and extracting meaningful information from them are very difficult tasks. The objective of this project is for a computer program to learn how to tell the time on an image of an analog clock.

In an image with large area and multiple objects, a machine would have to recognize the pattern of a clock and its location in the image. That alone is quite challenging, but it will be skipped here. The focus here is to tell the time so images contain only a clock. In order to read a clock's time, a person/machine must identify the hand positions and differentiate between the hour, minute, and second (if present) hands. The problem can decomposed into those two parts.

Object recognition and object interpretation fall into fields of image processing and computer vision. How can a machine emulate a human or animal's visual system? The best connection available is the 2-D Gabor filter [1], which originates from Dennis Gabor's 1-D filter (1946). 2-D Gabor filters accurately model the receptive field profiles of simple cells in the visual cortex [1]. 2-D Gabor filters have the mathematical form of a 2-D Gaussian multiplied by a 2-D complex exponential (see equations below). Both spatial and frequency domain versions have the same form with parameters interchanged [1].

Gabor filter in spatial domain:

$$\begin{aligned} \psi(x, y; f_0, \theta) &= \frac{f_0^2}{\pi\gamma\eta} e^{-\frac{f_0^2}{\gamma^2}x'^2 + \frac{f_0^2}{\eta^2}y'^2} e^{j2\pi f_0 x'} \\ x' &= x\cos\theta + y\sin\theta \\ y' &= -x\sin\theta + y\cos\theta \end{aligned}$$

Gabor filter in frequency domain:

$$\begin{aligned} \Psi(u, v; f_0, \theta) &= e^{-\pi^2 \frac{u'-f_0}{\alpha^2} + \frac{v'}{\beta^2}} \\ u' &= u\cos\theta + v\sin\theta \\ v' &= -u\sin\theta + v\cos\theta. \end{aligned}$$

The parameters of the Gabor filter control orientation, frequency scale, bandwidth, and aspect ratio [2]. There are many variations of Gabor filters in the literature, but they have the same basic form. An example of a filter bank of Gabor filters is shown in Figure 1. The filter bank consists of 5 frequency scales (exponentially spaced) and 8 orientations (linearly spaced).

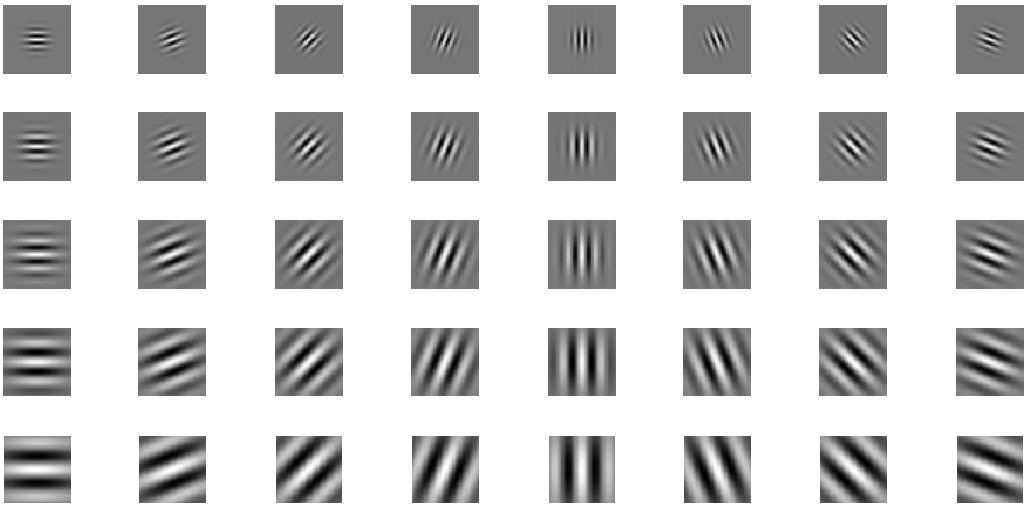

Figure 1. Real part of Gabor filter bank consisting of 5 frequency scales (exponentially spaced) and 8 orientations (linearly spaced)

The orientations of the Gabor filters are clearly visible in Figure 1. By convolving each Gabor filter with an image, features of an image can be extracted. The convolution equation is shown below.

$$r_\xi(x, y; f, \theta) = \psi(x, y; f, \theta) * \xi(x, y)$$
$$= \iint_{-\infty}^{\infty} \psi(x - x_\tau, y - y_\tau; f, \theta)\xi(x_\tau, y_\tau)dx_\tau dy_\tau$$

An example of the filter responses after convolution (with image in Fig 2) is shown in Figure 3.

Figure 2. Example image for Gabor filter response
(source: http://stackoverflow.com/questions/20608458/gabor-feature-extraction)

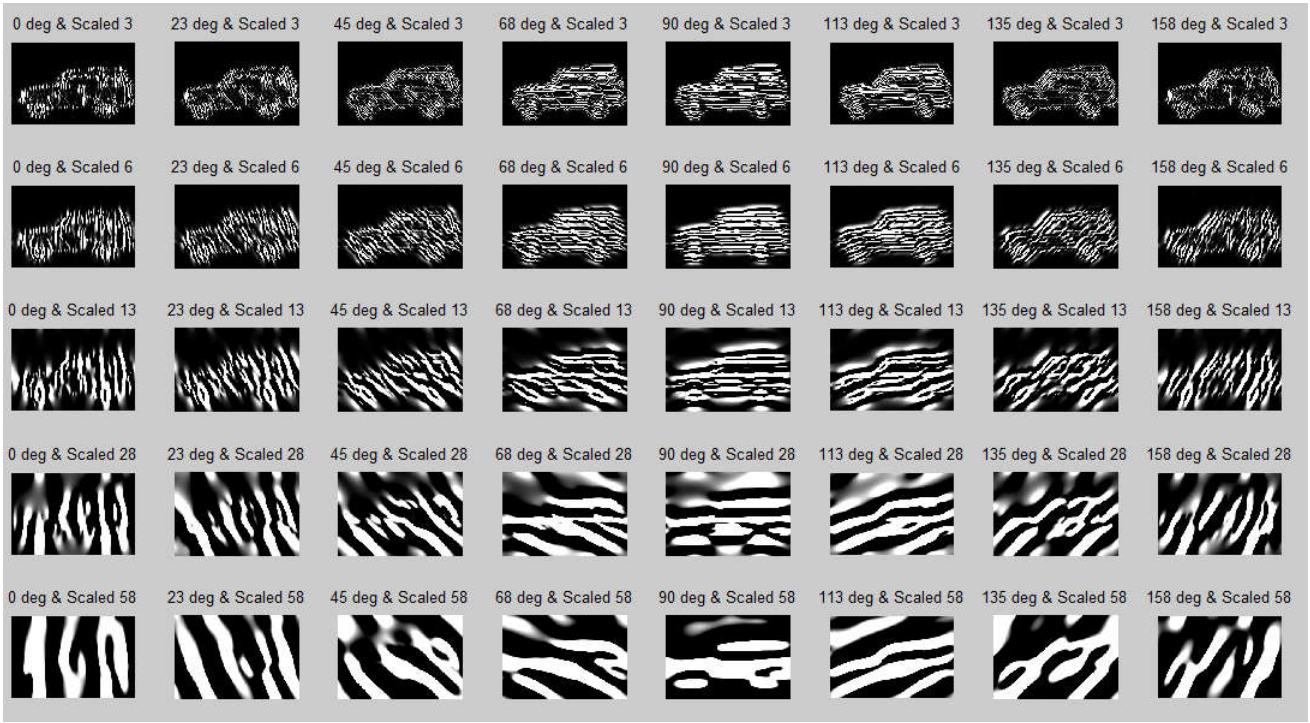

Figure 3. Absolute value of filter responses after convolution of Gabor filter bank with image in Figure 2.
(source: http://stackoverflow.com/questions/20608458/gabor-feature-extraction)

The same filtering can be achieved using convolutional neural networks (CNN), which are a type of deep learning neural network. The convolutional layers implement Gabor filters. Some examples of CNNs are LeNet-5, HMAX, and NeoCognitron [3]. Two publicly available implementations of CNNs are Theano[1] and *cuda-convnet*[2] that can utilize GPU computing. CNNs take advantage of the strong spatial correlation in images to parallelize the computations and avoid fully connected layers until near the output [3]. CNNs have proven accurate performance on many datasets of images.

1. http://deeplearning.net/software/theano/
   http://deeplearning.net/tutorial/lenet.html

2. https://code.google.com/p/cuda-convnet/

Methods and Results

Many approaches were attempted to achieve the objective. Since the Theano Python library is available, it was installed along with its dependencies (very complicating). Unfortunately, the Theano library is very complex and has a steep learning curve. Even attempts to modify the example code for the MNIST dataset proved too time consuming. However, the possibility of accelerating computation speed with the GPU was very compelling.

The next approach was to create a Gabor filter bank directly (via code), convolve the filters and images, and then extract useful features for a neural network (NN), support vector machine (SVM), or other form of machine learning. After reading some seminal papers in the field [1], [4], it became clear that parameter selection for Gabor filters is not trivial. Also, details of implementation, illustrative examples, and pseudocode are not present in most of the published papers.

The application of Gabor filters to iris recognition in [4] is amazing; it is very similar in nature to this project. Both irises and clocks are circular so object recognition would be about the same. However, irises are often partially occluded by eyelids, and they have far greater detail than most clock faces. Therefore, locating irises on a person's face and extracting features from them seems significantly more difficult than reading time on a clock. The researcher, John Daugman, commercialized the iris recognition technology and did not include many of the details of his implementation and algorithms. The concept of locating each iris, creating a "doubly dimensionless projected polar coordinate system" (see Figure 4), remapping the Cartesian pixels to the polar region, and extracting features using Gabor filters is absolutely relevant.

Daugman computes the sign (for real and imaginary parts) of projections of local regions of the remapped iris image on 2-D Gabor filters. The signs of real and imaginary parts are quantized as two bits, which represent the quadrant of a phasor in the complex plane (see Figure 5). The phase is encoded while the amplitude is not used. Phase captures wavelet zero crossings [5] and more detailed feature information than amplitude [6].

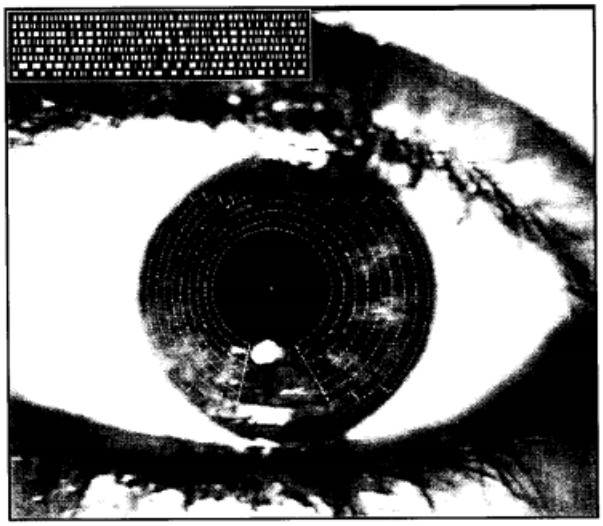

Figure 4. Example of locating each iris and creating a "doubly dimensionless projected polar coordinate system" on the iris, iris code is visible in upper left, excerpted from [4, p1152].

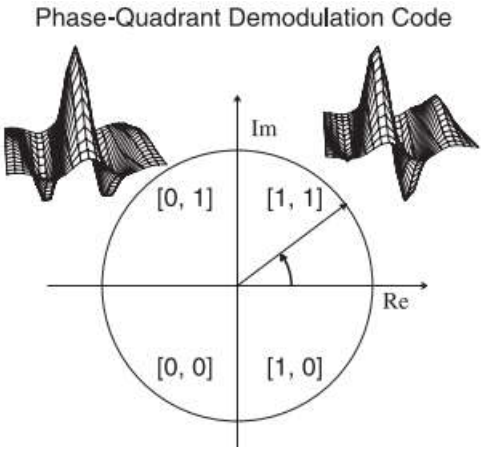

Figure 5. "Phase demodulation process used to encode iris patterns (excerpted from [5, p10])."

While Daugman's approach to iris recognition would seem to be well suited and yield similar results for the clock time problem, he omits too much information to figure out in a limited amount of time for this project. It is noted that comparison of iris codes and statistical testing was not the planned approach for evaluating/interpreting the features extracted from the clock images. Machine learning is applied to learn patterns from the clock features.

Relevant clock features should distinguish positions of clock hands and differences between the hands present. An analog mechanical clock has continuous angular movement. For this project,

we discretize that movement into 12 hour positions and 60 min/sec positions. As a result, there are 12 x 60 = 720 combinations of hour and minute hands (second hand is neglected). Using macro recording software, 720 screenshots of those combinations were captured for the Windows 7 built-in clock. Those screenshots were then processed: cropped to clock region, thresholded, and converted to black & white. The clock images are centered with normal 12:00 noon vertical orientation. An example clock is shown in Figure 6.

Figure 6. Example clock image (172x172 pixels) after processing from Windows 7 built-in clock.

With a CNN, the image could be input directly and the CNN will filter and then extract features on its own (with good architecture and coding of course). With features extracted, the CNNs fully connected layer(s) can ideally interpret the features. Using Gabor filters implemented in code, the Cartesian image can be convolved with each filter, yielding a distinct response. Convolution and Gabor filters' computation-intensive nature are their main disadvantage.

However, how best to use the filter responses is not entirely clear (at least for inexperienced people). For example, the magnitude (absolute value) filter responses of the image in Figure 6 and a Gabor filter bank with default parameters is shown in Figure 7. The filter responses where the clock hands coincided (or at least approximately so) with the filter's orientation are clearly visible. The filter response contains useful information spatially, but the responses are large in size (note higher frequency scales can be downsampled). From one initial image, multiple filter responses are obtained. For some applications such as texture analysis, the mean and standard deviation of each filter response can capture meaningful information; also, the local energy from summing the filter response of each orientation can be useful [7].

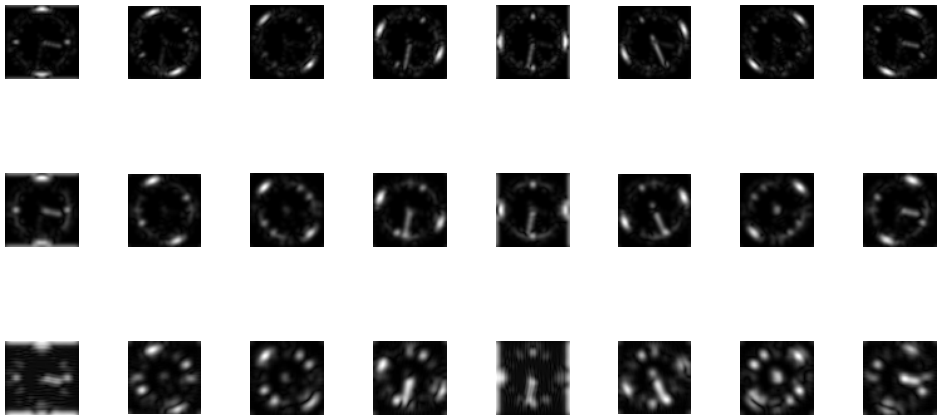

Figure 7. Magnitude (absolute value) filter responses of the image in Figure 6 and a Gabor filter bank with default parameters.

For the clock problem, mean, standard deviation, and local energy are not rich in discriminating features. That is a consequence of losing the phase information contained in the spatial location of features within the filter responses. It is not obvious how to extract the phase information in a compact form.

With the project's time running out and no clear way to use the filter responses effectively, another simpler approach was attempted. First, the Cartesian image is remapped to a polar coordinate system using classical transform equations. The (x,y) coordinates are rounded (nearest neighbor within image boundaries) when the polar coordinates do not match. The transformed image is saved in a rectangular array where rows correspond to radius (r in pixels) and columns correspond to angle θ (from [0,2π]). Some of the original image is lost in the transform. A graphical example of a transformed clock is shown in Figure 8.

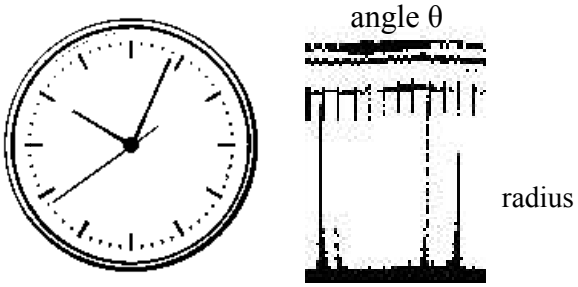

Figure 8. Original clock image (172x172 pixels) and transformed image to polar coordinates (86x60 pixels). Angle θ was sampled every 6° or π/30 radians. Radius units are per pixel.

This transformation is very similar to but not as sophisticated as Daugman's method in [4]. After transforming an image, the idea was to use the columns of the polar image matrix as feature vectors for a neural network with 86 inputs and 3 outputs (one each for hour, minute, and second hands). This NN would be trained with a subset of the 720 images until at least 99% accuracy was achieved. Then 60 copies of the NN would be assembled in parallel for new images; each copy would be wired to one column of the transformed image's polar matrix. The justification for training one and making 60 copies is from the clock's symmetry and greatly reduced computation time. There would be one NN copy for each minute/second position on the clock, and each NN could differentiate between the three hands. The clock's time corresponds to the three active outputs (or less if hands are overlapping).

The idea of 60 copies of one trained NN seemed plausible and functional. However, after designing a test/development NN and training it on 120 images for 10 epochs, accuracy was 0% and no progress was evident. Maybe it could work, but more likely the feature vectors are not good and class distinction is weak. Also, the hand positions in the columns of the polar image matrix tend to overlap on the side columns.

Another feature vector can be obtained from the polar image matrix by summing the columns. The information contained is much more visible after normalizing the vector by subtracting its mean and dividing by its standard deviation. A graphical example is shown in Figure 9.

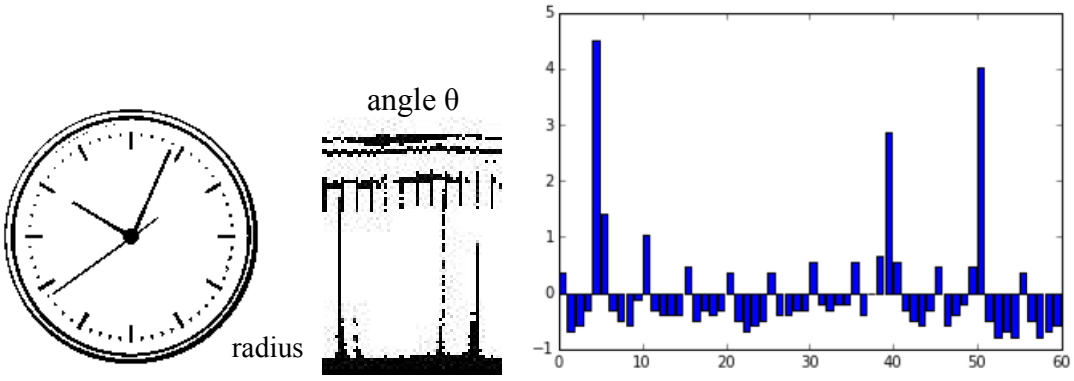

Figure 9. Bar graph of normalized feature vector from the polar image matrix (obtained by summing the columns and normalizing). Clock hand positions are clearly visible.

In the summed column feature vector (Figure 9), the positions of the clock hands correspond to the three bars with greatest amplitude. A machine learning algorithm would have to learn to

differentiate between the hands and output their positions. Time has run out for this project, but it seems entirely possible. The main disadvantage with the approach is that it's not very general. The original image had to be processed extensively, and even after that, it's not rotation invariant (however, the clock does not have a 12:00 marker anyways). Except for losing resolution from Cartesian to polar conversion, an advantage is that the method is scale invariant.

Conclusion

Machine learning for image processing, and image interpretation was explored in journal papers and websites for open source projects. An attempt at using a convolutional neural network was made with the Theano Python library; however, the library's complexity was too high to learn in a limited timeframe. A great deal of research on Gabor filters has been published in journals, and some seminal ones were reviewed. Again, the complex details of Gabor filter parameters became too difficult for the project timeframe. Sample clock images were convolved with Gabor filters and their responses obtained. The filter responses appeared to contain useful information, but it was not clear how to extract those features in a compact form. Code to transform Cartesian images to a polar representation was successfully implemented. Columns of the polar image matrices were used for training and testing of a neural network in order to output the hour, minute, or second hand of the clock. Unfortunately, the NN showed no progress and no accuracy. Another feature vector was obtained from the polar image matrix, and it clearly showed the positions of the clock hands and differences in their amplitude. That feature vector had potential to yield accurate results from a NN or other learning algorithm, but the overall approach was not very general as in a CNN. A general architecture for object recognition and image interpretation continues to be sought after.

References

[1] Daugman J., "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression," IEEE Trans on Acoustics, Speech, and Signal Processing. Vol. 36. No. 7. July 1988.

[2] Ilonen, J., Kamarainen, J.-K., Kälviäinen, H., "Fast Extraction of Multi-Resolution Gabor Features," 14th International Conference on Image Analysis and Processing, 2007.

[3] Theano,

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley and Y. Bengio. "Theano: A CPU and GPU Math Expression Compiler". Proceedings of the Python for Scientific Computing Conference (SciPy) 2010. June 30 - July 3, Austin, TX.

F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio. "Theano: new features and speed improvements". NIPS 2012 deep learning workshop.

http://deeplearning.net/software/theano/
http://deeplearning.net/tutorial/lenet.html

[4] Daugman J., "High Confidence Visual Recognition of Persons by a Test of Statistical Independence," IEEE Trans on Pattern Analysis and Machine Intelligence, Vol. 15, No. 11, November 1993.

[5] Daugman J., "Recognising Persons by Their Iris Patterns," *Advances in Biometric Person Authentication*, Springer-Verlag Berlin Heidelberg 2004.

[6] Oppenheim A., Lim J., "The Importance of Phase in Signals," Proceedings of the IEEE 69, pp. 529-541, 1981.

[7] Arivazhagan S., Ganesan L., Padam Priyal S., "Texture Classification Using Gabor Wavelets Based Rotation Invariant Features," Pattern Recognition Letters 27, 2006.